IDENTIFICATION AND RECONSTRUCTION OF BULLETS FROM MULTIPLE X-RAYS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF SCIENCE

AT THE UNIVERSITY OF CAPE TOWN

IN FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Simon Perkins

June 2004

Supervised by

Dr Patrick Marais

Acknowledgements

My heartfelt thanks go to the following:

My Supervisor
My Family
The Pinelands Gang (you know who you are)
The Roslyn Road Commune
The Denizens of the CVC Lab

who each in their own way contributed to this dissertation and without whom it would have been impossible to complete.

"Towards thee I roll, thou all-destroying but unconquering whale; to the last I grapple with thee; from hell's heart I stab at thee; for hate's sake I spit my last breath at thee"
— Captain Ahab to the Great White Whale

Moby Dick by Herman Melville

Abstract

The 3D shape and position of objects inside the human body are commonly detected using Computed Tomography (CT) scanning. CT is an expensive diagnostic option in economically disadvantaged areas and the radiation dose experienced by the patient is significant.

In this dissertation, we present a technique for reconstructing the 3D shape and position of bullets from multiple X-rays. This technique makes us of ubiquitous X-ray equipment and a small number of X-rays to reduce the radiation dose. Our work relies on Image Segmentation and Volume Reconstruction techniques.

We present a method for segmenting bullets out of X-rays, based on their signature in intensity profiles. This signature takes the form of a distinct plateau which we model with a number of parameters.

This model is used to identify horizontal and vertical line segments within an X-Ray corresponding to a bullet signature. Regions containing confluences of these line segments are selected as bullet candidates. The actual bullet is thresholded out of the region based on a range of intensities occupied by the intensity profiles that contributed to the region.

A simple Volume Reconstruction algorithm is implemented that back-projects the silhouettes of bullets obtained from our segmentation technique. This algorithm operates on a 3D voxel volume represented as an octree. The reconstruction is reduced to the 2D case by reconstructing a slice of the voxel volume at a time.

We achieve good results for our segmentation algorithm. When compared with a manual segmentation, our algorithm matches 90% of the bullet pixels in nine of the twelve test X-rays. Our reconstruction algorithm produces an acceptable results: It achieves a 70% match for a test case where we compare a simulated bullet with a reconstructed bullet.

Contents

List of Figures

Chapter 1

Introduction

Due to the prevalance of violent crime in South Africa, a large percentage of trauma patients admitted to hospitals suffer from bullet wounds. The bullets that caused these wounds are sometimes lodged within the body. It is important to identify the location and shape of these bullets within the body for the purpose of surgical planning.

In order to identify this location, an internal scan of the body must be performed. *Computerised Tomography* (CT) [23] is by far the most prevalent method of obtaining a three-dimensional representation of the internal structure of the body. CT has been in used since the early 1970's.

In economically disadvantaged regions CT remains an expensive option. In South Africa where the national health care budget is stretched by low resources and a burgeoning population, CT Machines are expensive for hospitals to buy, operate and maintain. Only major health care centres can afford this equipment. The costs associated with CT are often passed to the patient, who is usually unable to afford this diagnostic treatment.

Additionally, to generate an accurate representation of the internal structure of the body, CT needs to project a large number of X-rays through the body from a wide range of angles. Therefore, the radiation dosage received by a patient is also significant.

Due to the logistical and radiation issues that have been mentioned, it would be advantageous to provide an alternative solution to CT diagnostic treatment when searching for bullets lodged in the body. This solution would therefore need to be relatively inexpensive, have a wide geographic distribution and reduced radiation dose.

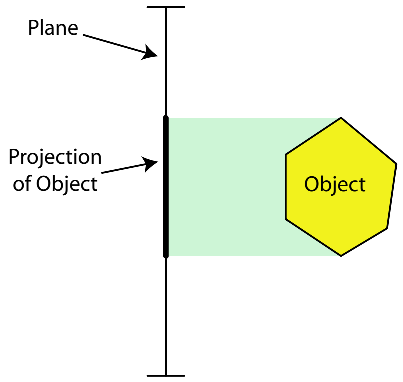

Figure 1: Projecting a 2D object onto a 1D plane.

The X-ray scanner was the first machine used to perform internal scans of the structure of the body. It has been in use for over a century now. Most rural clinics have an X-ray machine and they are relatively inexpensive to operate. However, a X-ray machine only produces a *two-dimensional* projection of a three-dimensional structure. Fortunately, with the proper application of Computer Vision techniques it is possible to obtain a three-dimensional approximation of the of the structure that is being scanned.

1.1 Volume Carving

The Computer Vision technique of *Volume Carving* [2] is used to approximate a three-dimensional object from a number of two-dimensional projections of the object taken at multiple angles. Mathematically speaking, a projection maps a function operating in a space of dimension n to a dimension $n - 1$. Figure 1 shows how an object is projected from a two-dimensional space onto a one-dimensional space or plane. Figure 2 indicates how a two-dimensional object can be approximated from a number of one-dimensional projections.

Therefore, given enough X-rays of an object it is possible to generate an approximation of the shape of the object. The problem with using the Volume Carving technique on an X-ray of a patient with a bullet wound is that many other bodily structures are projected as well. Thus, an X-ray image will show the bullet as well as the bones and flesh of the patient. In order for Volume Carving to work,

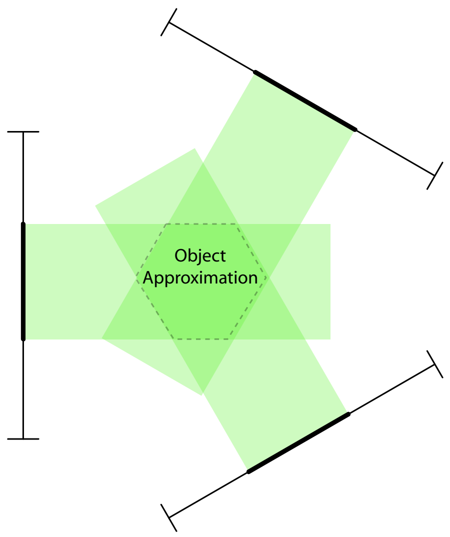

Figure 2: Approximating a 2D object from a three 1D projections

only the projection of the bullet can be used to approximate its three-dimensional shape. To deal with this problem we will need to turn to the field of *Image Segmentation*.

1.2 Image Segmentation

The region occupied by the bullet in the X-ray must be distinguished from other regions in the X-ray image. This process of distinguishing and identifying regions within an image is referred to as *Image Segmentation* [43]. Image Segmentation is a very broad field with many different techniques and approaches. Often it is necessary to specifically develop a technique that is tailored to identify the specific properties that a region exhibits within an image. This can be difficult if these properties do not differ significantly from surrounding regions. Fortunately, bullets exhibit distinct properties in X-ray images: they form regions of fairly uniform intensity.

1.3 Research Aims

From the topics that we have briefly presented, it is plausable to suggest that using the techniques of *Volume Carving* and *Image Segmentation*, it is possible to approximate the three-dimensional shape and position of a bullet from a number of X-rays taken multiple angles. This hypothesis forms the basis of our research.

The overall aim of our research is to develop a cheap three-dimensional imaging technique that can replace CT when searching for bullets within the body. As mentioned in the previous section, we will need to investigate two areas of Computer Vision to accomplish this.

1. **Image Segmentation**: We need to develop a segmentation algorithm to identify bullets within X-rays. This algorithm should be as automatic as possible. i.e. it should require very little user intervention.
2. **Volume Carving**: We need to develop a reconstruction technique that recovers the three-dimensional shape and position of the bullet from multiple X-ray projections. The reconstruction field is very mature. Therefore we do not need to research a new technique for performing reconstruction. However, we will need to implement a suitable technique in order to prove that the method that we are proposing works.

Using these techniques it should be possible to provide an alternative to CT that can be implemented on cheap, widely available X-ray machine equipment. There are calibration issues associated with this technique that involve making decisions on how to align the different X-rays that are input to the volume carving algorithm. This is a closed problem and is not addressed in this dissertation.

The radiation dose administered using our proposed technique should also be lower than that of a CT scan. The reason for this is that CT makes no use of a priori information regarding the data that is being reconstructed. It is a general technique used for reconstructing the entire contents of a volume and therefore it requires many X-ray projections to accurately approximate the volume.

Our proposed technique searches for an object within an X-ray image and uses this information to infer the three-dimensional shape of the object. Therefore, we use a priori information about the object we are reconstructing. This reduces the number of X-ray projections that are required to estimate an object significantly. Due to this reduction in the number of X-ray projections, our technique will result in less radiation being used to perform a reconstruction.

1.4 Overview of Dissertation

The framework of this dissertation is as follows

- **Chapter 2: Background.** We present some of the relevent background information that is central to this dissertation. The properties of X-ray radiation are discussed. We also present an overview of a number of segmentation techniques that are widely used in the field of Computer Vision. Finally we discuss reconstruction of three-dimensional objects from two-dimensional projections. We discuss two techniques from this field, *Volume Carving* and *Computed Tomography*.
- **Chapter 3: Segmentation.** In this chapter we present the algorithm that we use to segment bullets in X-ray projections. Firstly we discuss the properties of bullets and proceed to derive a model from these properties. We then describe how we use this model to develop the segmentation algorithm.
- **Chapter 4: Volume Reconstruction.** This chapter describes the process we use to reconstruct the three-dimensional shape and position of a bullet from multiple X-ray projections. Firstly, we discuss the data structure that we use to represent our volume. Secondly, we discuss the image and projection data that is input to the reconstruction algorithm. Finally we describe the reconstruction process itself.
- **Chapter 5: Evaluation.** We present the results of our research in this chapter. The segmentation and volume reconstruction sections of our project are tested separately. We describe the tests that we use to obtain these results and discuss the implications of our results. We also describe the X-ray simulator that we use to generate artificial X-rays for testing our reconstruction algorithm.
- **Chapter 6: Conclusion and Future Work.** Finally, we conclude our work, describing what we have achieved. We also mention possible avenues for future work.

Chapter 2

Background

This chapter describes the background material that is related to our work. We first examine X-Rays and X-Ray images in order to provide an understanding of how their properties in the physical world are used to generate images.

Secondly, we discuss the topic of *Segmentation*, the identification of regions with in an image. This section describes a number of different techniques that are used to identify structures within an image. We will cover basic segmentation techniques, the Canny Edge-Detector, the various Snake algorithms and Seeded Region Growing algorithms.

Finally, we introduce the reconstruction of three-dimensional structures from two-dimensional images. This class of algorithms estimates a three-dimensional object from its projections. In this section we cover *Inferred Visual Hulls* and *Computed Tomography*. Three-dimensional reconstructions are very useful. For example, Computed Tomography scans generate a three-dimensional representation of the internal structures of the human body.

2.1 X-Ray Images

X-Rays are a form of electromagnetic radiation which can be used to perform examinations of the internal structures of the human body. X-Ray images are used to diagnose a wide range of medical conditions such as bone fractures and breast tumours. This section will describe the physical properties of X-Rays and their creation. We will also describe how an X-Ray machine operates and how X-Rays images are created.

2.1.1 X-Rays

X-Rays are a form of electromagnetic radiation carried by photons [44]. X-Rays have an extremely short wavelength: 10 nanometres and below. One of their most useful properties is their ability to penetrate objects that are opaque to visible light. Both visible light photons and X-Ray photons are generated by the movement of electrons within an atom. An atom consists of a number of electrons situated at different energy levels around it's nucleus. In order for an electron to drop to a lower energy level it needs to release some energy. It does this by releasing a photon. The energy level of the photon is dependant on the number of energy levels that the electron dropped.

Atoms do not normally emit radiation. However, if a fast-moving electron strikes an atom, it may collide with one of the atom's electrons and push it up to a higher energy level. However, the electron is unstable in its new state and falls back down, releasing the energy as an electromagnetic wave in the form of visible light.

A similar process occurs in the creation of X-Rays. However, in this case both the colliding electron and the electron that struck it carome off. An electron from a higher energy level immediately replaces the lost electron. This shift in energy level is very abrupt since the attraction between the nucleus and the electron is much greater. Correspondingly, the wavelength of the electromagnetic wave that is released is much shorter, less than 10 nanometres. These waves are X-Rays.

The X-Rays with the highest frequency (and therefore the shortest wavelength) are generated when free electrons strike the nucleus of an atom. These collisions are not fully understood. It appears that when a electron strikes the nucleus of an atom head-on, the field of the atom stops the electron dead in its track and converts the mass of the electron into an X-Ray of very high frequency and wavelength. Other electrons are deflected by the nucleus and decelerate. This generates an emission of energy proportional to the decelaration factor.

Therefore, when an atom with a large nucleus is bombarded by a stream of electrons, two ranges of X-Rays are created. Firstly, X-Rays created from the collision of free electrons with orbital electrons in an atom's nucleus. Their wavelength and energy correspond to the energy levels that the electrons occupy. Secondly, there are X-Rays created from the collision of free electrons with the nucleus of an atom. These X-Rays occupy a wavelength ranging from the ultraviolet to the infinitesmal.

X-Ray Images

X-Rays where first discovered by Wilhelm Roentgen in 1895. One of the first experiments that Roentgen performed was to take an X-Ray image of his wife's hand by placing the hand between an X-Ray source and some photographic film. The X-Ray machine still follows the same basic principles today.

The most important component of an X-Ray machine is the cathode anode electrode pair that is placed within a sealed glass vacumn tube. A direct-current potential is created between the cathode and the anode. This creates an electric field. Positive ions are driven by this field to bombard the cathode, releasing electrons. Due to the large potential difference created between the cathode and the anode, these electrons are attracted to the anode. The anode is usually made of tungsten. As explained earlier, the collision between the tungsten atoms and the electrons generate x-rays. The X-Rays that are released are focused through a filter to form a beam of X-Rays.

The object that is being X-Rayed is placed between the X-Ray source and a sheet of photographic film. The direct current is created between the x-ray sources cathode-anode pair, bombarding the photographic film with x-rays. The object in between the source and the film scatters the stream of x-rays depending on its density. Flesh will allow most X-Rays to pass through it since it consists largely of water which is a very a light molecule. Bone is denser and causes more X-Ray scattering. When an object causes X-Ray scattering, the photographic film behind it is exposed to less X-Ray radiation and therefore will not fade to black as quickly as the parts of the film that are not exposed to radiation. Thus, an X-Ray image is created, representing the amount of scattering experienced by the X-Rays as they travelled to the photographic film. An X-Ray image can also be thought of as an image of density.

2.2 Segmentation

Segmentation, in the context of imaging, is the identification of important regions within an image. It falls under the general category of "Computer Vision", the science of developing algorithms that allow a computer to perceive and understand visual information [43]. Segmentation is an especially important field of Biomedical Imaging.

For example, in [34], the bones of a patient's hand are segmented out of a set of x-rays. These regions are then used to measure whether the skeleton of the patient has experienced growth that

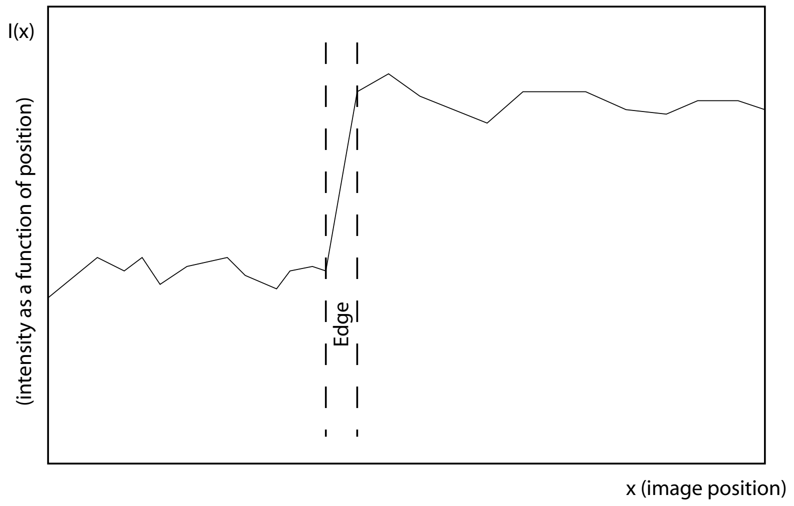

Figure 3: This graph shows the intensity of an image as a function of its position. The section of the graph where the value of the image intensity changes rapidly is marked as an edge

is accelerated or retarded with respect to the patient's age. This system is helpful in diagnosing conditions such as diabetes and arthritis.

In this section we cover some of the most common techniques used in Medical Segmentation. We first cover thresholding and edge-detection techniques and then move on to the more advanced methods of edge-detection and region-growing.

2.2.1 Basic Segmentation Techniques

One of the simplest segmentation techniques is the *Thresholding* [43] algorithm. This algorithm is based upon the idea that regions of an image will occupy a range of intensities between an upper bound and a lower bound. Thresholding retains these intensities while disposing of those outside the range. So for each point p in an image $I(x, y)$, the image $T(x, y)$ obtained by thresholding between the lower bound l and the upper bound u, is defined by

$$T(p) = \begin{cases} 1 & \text{if } l \leq I(p) \leq u \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

For this algorithm to work the exact range of intensities that the region occupies needs to be known. These algorithms ignore the concept of spatial locality: separate regions may be lumped together because they share the same range of intensities.

Other algorithms are based on the idea that pixel values will change rapidly at a region boundary. The original image is filtered with a Sobel or a Roberts [43] filter. These filters produce large values at parts of the image where the intensity values change rapidly. The sections of high intensity are then processed to produce a continuous edge representing the boundary of the region. This concept is illustrated in Figure 3. For this reason, these algorithms are called *Edge Detection Algorithms*. These algorithms suffer when the boundary of a region is not distinct since a number of different edges may be produced that represent part of the boundary of the region. These edges then need to be joined which can be difficult if there are edges from other regions nearby.

2.2.2 Edge Detection - Canny

One of the best-known edge detectors developed to date is the *Canny* edge detector [9]. Canny developed his edge detector to satisfy three performance criteria

Good Detection: There should be a low probability of failing to mark real edge points, and low probability of falsely marking non-edge points.

Good Localization: The points marked as edge points should be as close as possible to the true centre of the edge.

Only One Response to a Single Edge: If there are two responses to an edge, one of them must be false. This criterion is similar to the first.

There are a number of steps to the Canny Edge Detection process. Firstly, the input image is smoothed using a Gaussian convolution. This convolves the image signal with the bell-shaped Gaussian filter in order to remove noise from the signal.

The following simple two-dimensional first derivative operator is then applied to the smoothed image.

$$M(x,y) = \left(\frac{\partial I}{\partial x}^2(x,y), \frac{\partial I}{\partial y}^2(x,y)\right) \tag{2}$$

where I is the original image and x and y are its coordinates. This operator highlights the portions of the image with a high first derivative. The image that results from this process is the *gradient magnitude image*. Edges in the original input image cause ridges in the gradient magnitude image. These ridges are tracked and thinned using a process called *non-maximal suppression* to form edges that are one pixel thick.

The tracking process is controlled by two threshold levels α and β where $\alpha > \beta$. Tracking may only begin on a ridge at a pixel value greater then α, but will continue in both directions until the pixel value falls below β. This process is called *hysteresis* and helps to prevent noisy edges from being split up into multiple edges.

The output of the Canny Edge Detector is determined by three parameters, the width of the Gaussian filter and the the upper and lower threshold levels, α and β. Increasing the width of the Gaussian mask will reduce the edge detector's sensitivity to noise at the expense of some of the finer detail in the image.

Generally, setting the upper threshold level α to a high value and the lower threshold β to a low value will produce good results. Setting the β too high will cause noisy edges to break up and setting α too low will make the detector find too many small edges or edge fragments that are often unnecessary.

One of the problems associated with the Canny Edge Detector is its difficulty in dealing with Y-junctions [9]. This occurs when three ridges meet each other at a single point. The detector treats two of the ridges as one line and the third as a line that almost joins up with the other two.

2.2.3 Snakes

In this section we cover the so-called "Snake Algorithms". We will describe the original *Active Contour Model* [24] developed by Kass et al. and then examine two techniques that were derived from it, *Pressurised Snakes* and *Active Region Models*.

Active Contour Models

An Active Contour Model [24] or snake is a parametric contour that deforms over a series of time steps or iterations [20] due to the minimisation of its energy. The contour, u, depends on two parameters, s which varies from 0 to $N-1$, and t which is the current iteration in time.

$$u(s,t) = (x(s,t), y(s,t)) \begin{cases} s = \text{spatial parameter} \\ t = \text{time parameter} \end{cases} \tag{3}$$

The energy of the contour is determined by a number of forces that are classified as *internal*, *external* and *image* forces. The sum of the energies determine the total energy of the snake

$$E_{\text{snake}} = \int_0^{N-1} E_{\text{internal}}(\mathbf{u}) + \int_0^{N-1} E_{\text{external}}(\mathbf{u}) + \int_0^{N-1} E_{\text{image}}(\mathbf{u}) \tag{4}$$

Internal forces control the stiffness and tension of the contour model and ensure that the contour remains smooth and does not curve too sharply. $\alpha(s)$ and $\beta(s)$ are user controlled constants.

$$E_{\text{internal}}(u) = \underbrace{\alpha(s)\left|\frac{\partial \mathbf{u}}{\partial s}\right|^2}_{\text{Tension}} + \underbrace{\beta(s)\left|\frac{\partial^2 \mathbf{u}}{\partial s^2}\right|^2}_{\text{Stiffness}} \tag{5}$$

External forces are high-level forces that are introduced by the user in order to make the snake behave in a certain way. An expansion parameter can be inserted in order to encourage the snake to balloon outwards or a spring force can be created between a point $\mathbf{i}$ and a point on the snake $\mathbf{u}$, by using the following equation for the external energy term

$$E_{\text{external}}(\mathbf{u}) = k\,|\mathbf{i} - \mathbf{u}| \tag{6}$$

Image forces are generated from processing an image in order to identify points of interest that the snake will move toward. For example, snakes are most commonly used as edge detectors. The following energy equation causes the snake to be attracted to edges. G_σ is a Gaussian filter with standard deviation σ.

$$E_{\text{image}}(\mathbf{u}) = -\left|\frac{\partial}{\partial \mathbf{u}} G_\sigma * I(\mathbf{u})\right|^2 \tag{7}$$

By minimising the energy of the snake, the snake is encouraged to move itself to find image features. Minimising the tension term causes the snake to contract, while minimising the stiffness term prevents the snake from bending too much. Minimising the edge term in equation 7 causes the snake to be attracted to the minimums created by strong edges in the image. In practice, the snake is moved towards a local minimum using an iterative gradient descent method such as *Euler Time-Stepping* or *Implicit Energy Minimisation*.

The basic snake algorithm uses edge energy to drive the segmentation. This means that the algorithms performance depends largely on the quality of edges obtained from an image. If there are many edges or the strength of the edge that is being searched for is weak compared to other edges, the snake will have difficulty obtaining a correct solution.

The *Active Shape Model* [14] is an extension of the Active Contour Model that attempts to deal with this issue by teaching a snake to conform to particular contour configurations. The snake algorithm is initialised with a set of training contours which it uses to seek out similar contours within an image. This extension of snakes is useful if the shape of the region that is being searched for is known beforehand.

Pressurised Snakes

As described in the previous section, minimising the tension term of a snake causes the snake to contract to a point. Cohen et al. therefore introduced the idea of a *Pressurised Snake* [13]. An extra pressure term is added to the snake energy equation.

$$E_{\text{pressure}}(\mathbf{u}) = -\rho(s) \int_0^{N-1} \frac{\partial \mathbf{u}}{\partial s} \times \mathbf{u} \, \mathrm{d}s \tag{8}$$

This causes the snake to expand in a direction perpendicular to the contour. The edges that a pressurised snake is searching for should be strong otherwise the snake may grow out of control.

Active Region Models

In the implementations of snakes that we have described, edge forces have been used to drive the snake towards features in an image. As we have noted, these edges have to be fairly strong otherwise the snake may not be sufficiently attracted to them. *Active Region Models* [21] dispose of the edge energy term and replace it with a region energy term. This region energy is a function of statistical properties of the pixels in the region enclosed by the snake, and generates a pressure that causes the snake to expand or contract to fit a homogenous region. This function is called the "Goodness" function G.

Therefore the snake will expand when the pixels within it lie within a statistical measure and contract when they are outside it. Statistical Distance Metrics are well-suited for use as goodness functions [21]. The energy equation of the snake now looks as follows.

$$E_{\text{snake}} = \underbrace{\alpha(s)\int_0^{N-1}\left|\frac{\partial \mathbf{u}}{\partial s}\right|^2}_{\text{TensionEnergy}} + \underbrace{\beta(s)\int_0^{N-1}\left|\frac{\partial^2 \mathbf{u}}{\partial s^2}\right|^2}_{\text{StiffnessEnergy}} \underbrace{- \rho\iint_R G(I(x,y))\,\mathrm{d}^2 A}_{\text{RegionEnergy}} \tag{9}$$

The function G measures the extent to which the pixels $I(x, y)$, within the region enclosed by the snake R, fall within the accepted statistical limits. G is designed in such a way as to produce large values for the areas that are being searched for. A pressure term associated with the region energy is used to determine the direction of the region pressure at a particular point on the snake. In the following equation f_{pre} is simply the normal to the boundary weighted by the local value of the goodness function.

$$f_{\text{pre}} = \frac{\rho}{2} G(I(\mathbf{u}))\left(\frac{\partial \mathbf{u}}{\partial s}\right)^{\perp} \tag{10}$$

A common choice for G is a linear function based on the mean μ and standard deviation σ of the region enclosed by the original configuration of the snake .

$$G(I) = 1 - \frac{1}{k\sigma}\left|I - \mu\right| \tag{11}$$

This function allows the pressure to reach an equilibrium when the boundary of the snake encounters pixels that fall on the statistical limit specified by μ and σ. k specifies how many standard deviations of the mean will be accepted. When $|I - \mu| = 0$ then the region will expand. Conversely when $|I - \mu|$ is large the region will contract very quickly. And when $|I - \mu| = k\sigma$ there will be no pressure.

Active Region Models perform well at segmenting homogenous regions and can be adapted to segment regions characterised by colour or texture. Ivins [21] demonstrated that an Active Region Model can segment a region of homogenous texture from an image containing a number of Brodatz Textures [6]. However, the texture algorithm parameters do need to be tweaked on a per image basis in order to obtain good segmentations, which implies a large degree of user control. The snake also needs to be initialised within the region that is being segmented. This is because the region mean needs to be representative of the region that is being segmented in order for the Goodness function to provide the correct pressure on the contour.

2.2.4 Region Growing

The Region Growing algorithm family focus on growing a region by adding connected pixels which fit certain criteria. This is based on the theory that the pixels contained in a region will be fairly homogenous. In this section we will examine *Seeded Region Growing* and then *Dynamic Region Growing*.

Seeded Region Growing

The *Seeded Region Growing* [1] algorithm segments an image based on the position and characteristics of a number of starting points or *seeds*. One of the goals of the paper was to produce a segmentation algorithm that is free of tuning parameters. The seeds are the only input to the algorithm and determine which parts of the image are classified as regions of interest or noise. Firstly, the seeds are grouped into n sets $A_1, A_2, \ldots, A_n$.

The algorithm is inductive and is based upon adding a pixel to one of the sets A_k at each step. Firstly, for each iteration, the pixels that neighbour with each set are grouped into a set T.

$$T = \left\{ x \notin \bigcup_{k=1}^{n} A_k | N(x) \cap \bigcup_{k=1}^{n} A_k \neq \emptyset \right\} \tag{12}$$

where $N(x)$ is the set of pixels that are neighbours to x. Then for each pixel $x \in T$, a distance metric $d(x)$ is calculated which measures how “far” x is from the region that it neighbours with. Firstly we define J, the set of the indices of the regions bordering x.

$$J = \{k | N(x) \cup A_k \neq \emptyset\} \tag{13}$$

The simplest metric compares x’s intensity with the mean intensity of the neighbouring region.

$$d(x) = |i(x) - \mathbf{mean}(A_j)| \tag{14}$$

where $j \in J$ and i returns the intensity of x. The region A_j which is x is added to is determined by finding the j which minimises $d(x)$. In other words, x will be added to the region whose mean intensity is closest to x's intensity value. This completes an iteration of the algorithm. The algorithm continues until $T = \emptyset$, there are no more boundary pixels to be found and the image has been tesselated into separate regions.

This algorithm was developed as a tool for performing manual segmentation and requires careful selection of the seeds in order to achieve good results. If a region exhibits noise, a range of seeds must be picked that represent the range of intensities within the region. This implies that a degree of user control is required in order to obtain a good segmentation. Adams proposes automating the algorithm by selecting local minima and maxima as seeds.

Dynamic Region Growing

Dynamic Region Growing [42] combines region growing with some features of edge-detection and is based on the theory that regions will have strong contours. As with *Seeded Region Growing* the input to the algorithm is a number of seeds. Siebert defines the *strength* of a region's contour $cs(R)$, as

$$cs(R) = \frac{1}{n} \sum_{p_i \in C_R} |p_i - q_i| \tag{15}$$

where C_R is the set of pixels on the contour of region R and $q_i \notin R$ are in the 4-connected pixel neighbourhood of p_i. n is the number of pixels on contour C_R. This metric provides a method of testing the quality of a region's segmentation.

Region Growing is performed by incorporating pixels that are connected to the region which satisfy the following condition

$$|\mu - I_p| < \epsilon \tag{16}$$

where μ is the mean intensity of the region, I_p is the intensity of the candidate pixel and ϵ is the distance from the mean that will be accepted. This algorithm iteratively grows a region by steadily increasing ϵ, the range of pixel intensities that will be accepted into the region.

Each region is initially defined by a seed pixel. At each iteration of this algorithm, region growing is performed based on ϵ and a region R_k is generated by adding pixels to R_{k-1}. When there are no more candidate pixels to be added to the region, $cs(R_k)$ is measured. ϵ is then increased and another iteration of the algorithm takes place, adding more pixels to R_k to produce R_{k+1}. This continues until an upper bound of ϵ, ϵ_{high} is reached or the region grows too rapidly, creating an overspill.

An overspill usually occurs when two regions of similar intensity are joined together, usually by a thin "bridge" of pixels. Siebert states that overspill is generally undesirable. Overspill can be detected by observing the following phenomena

- A rapid increase in the region's size occurs
- A decrease in the region's circularity
- A change in the region's average intensity

When ϵ_{high} is reached, the contour strengths $cs(R_k)$, of the region R during its growth are compared with one another to determine the point in its growth where its contour was strongest. This contour is then selected as the final segmentation of region R. The pseudocode for this algorithm is as follows

```
while seeds remain
        initialise $R_1$ with a seed
        $\epsilon = \epsilon_{low}$
        $k = 1$
        while $\epsilon \leq \epsilon_{high}$ and no overspill occurs in $R_k$
                compute contour strength $cs(R_k)$
                grow region $R_k$ to produce $R_{k+1}$
                $\epsilon = \epsilon + 1$
                $k = k + 1$
        end
        $R = R_j | cs(R_j) = max(cs(R_k))$
end
```

The values for ϵ_{low} and ϵ_{high} are not critical. ϵ_{low} can be set so that no reasonable segmentation can be expected for $\epsilon < \epsilon_{low}$. ϵ_{high} can also be set to a value beyond which reasonable segmentation is not expected, since the overspill condition will usually cause the algorithm to move to the next iteration before E_{high} is reached. If two regions overlap with each other, the larger region is kept, while the smaller one is thrown away.

In order to provide a level of automation in the Dynamic Region Growing algorithm, Siebert proposes a seeding strategy where seeds are constructed from micro-regions of similar intensity. Siebert notes that it is worthwhile being profligate with the number of initial seeds, since overlapping regions that inevitably occur from a large number of seeds will be dealt with as mentioned in the previous paragraph.

Dynamic Region Growing is a time consuming algorithm since a number of possible configurations of a region have to be tested before one is selected as a final region. The quality of the segmentation is also highly dependant on the seeds chosen, the only input to the algorithm. Automation is possible as suggested by the author, but this requires a large number of redundant seeds which further increase the time taken by the algorithm, even though the algorithm itself is linear in the amount of seeds.

2.3 Three-Dimensional Reconstruction from Two-Dimensional Projections

In this section we will introduce the techniques for generating a three-dimensional model from a number of two-dimensional projections. We will first introduce the concept of a visual hull, the estimation of a three-dimensional object from a number of images. We will then describe the method of Computed Tomography which generates a three-dimensional representation of the density of an object from x-ray projections.

2.3.1 Inferred Visual Hulls

One of the first techniques of volume reconstruction involved approximating the *visual hull* [4] of the imaged object. This technique is also known as *volume intersection*. The *visual hull* is defined as follows [2].

> The *visual hull* $VH(S, R)$ of an object S relative to a viewing region R is a region of the Euclidean space E^3 such that, for each point $P \in VH(S, R)$ and each viewpoint $V \in R$, the half-line starting at V and passing through P contains at least a point of S.

An intuitive definition is that the *visual hull* is the maximal object that gives the same silhouette of S from any viewpoint.

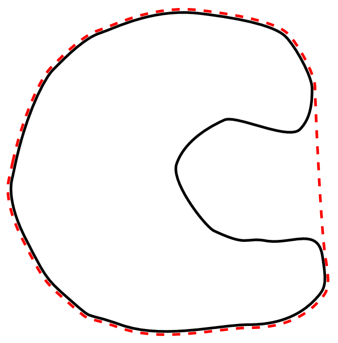

Figure 4: The Visual hull of an Object. The dashed line encloses the visual hull

The input to the algorithm is a set of N images that are projections of the object S onto N viewing planes. The projections of the object S must be segmented out of each image. These segmented regions are back-projected into 3D space and intersected. The resultant volume is the *inferred visual hull*. This can be seen in Figure 5.

An important property of the *inferred visual hull* is that the size of $VH(S, R)$ decreases as more images are used. However, this algorithm is not suited to concavities in the object S, since these concavities are usually occluded.

The first basic work in deriving a *visual hull* was accomplished by Martin and Aggarwal [30] in 1983. They dealt with the case of a number of orthogonal projections of an object. Firstly, the silhouette of the object in an image is discretised into a set of horizontal lines that describe the area that the silhouette encloses. The start and end points of these horizontal lines are back-projected to form a pair of lines that are coplanar and form a constraint on the extent of the approximated volume. When another pair of line contraints from another silhouette and which are in the same plane, are intersected with the original pair, a parallelogram is formed, finitely bounding the extent of the approximated volume. The locus of all the parallelograms in a plane is the *visual hull*.

Volume Representations

Martin and Aggarwal also developed a *volume segment* structure to efficiently represent the *visual hull*. This representation consists of a list of y-coordinate pairs which are accessed through a x-coordinate list (line), which is in turn accessed by a z-coordinate list (plane). This representation is far more space efficient than a three-dimensional voxel representation.

Chien and Aggarwal [11] experimented with using octrees to represent the *visual hull* of an object. An octree is a hierarchical structure used to represent a three-dimensional space. Octrees subdivide space by placing different sections of space into a node. Each node can have up to eight children, hence the name octree. If a node cannot exactly contain the space, the space is divided up among its children and the process continues until the representation is fine enough.

Their technique required three images taken from orthogonal viewing directions as input. The object in each image is converted into a quadtree representation which is subsequently merged into an octree. Unfortunately, the restriction of three images and the orthogonality requirement limit the quality of the reconstruction.

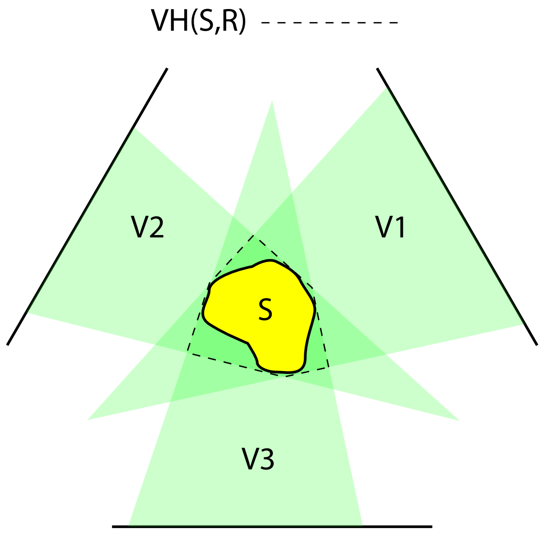

Figure 5: Deriving the Visual Hull $VS(S, R)$ of an object S from a number of viewpoints $V1, V2, V3$

2.3.2 Computed Tomography

Computed Tomography (CT) [23] is a technique for generating a three-dimensional representation of the density of an object. A CT machine consists of a number of x-ray emitters and detectors. The object for which the three-dimensional representation is being constructed is placed between the emitter and the detectors. The detectors measure how much the object attenuates the stream of x-rays projected by the X-Ray emitter and uses this information to generate the the three-dimensional representation.

Mathematically, x-rays are modelled as line integrals, the integral of some parameter of the object along a line. In this case line integrals are used to measure the total attenuation of an x-ray as it travels in a straight line through the object. Figure 6 will be used to describe line integrals and projections. In this diagram, an object is represented by the function $f(x, y)$. Also present in the diagram are a number of line integrals which are defined by the θ and t parameters. θ is the rotation angle of the projection. Rotating the (x, y) coordinate system by θ produces the (s, t) system. Using the coordinate systems defined in figure 6 a line can be defined as follows.

$$x\mathrm{cos}(\theta) + y\mathrm{sin}(\theta) = t \tag{17}$$

Using this line definition we define the line integral $P_\theta(t)$ over the object $f(x, y)$

$$P_\theta(t) = \int_{(\theta,t)} f(x, y)\mathrm{d}s \tag{18}$$

A projection is simply a set of line integrals. The simplest type of projection is a set of parallel line integrals as shown in figure 7 (a). A fan beam projection can also be contructed using a single fixed source relative to a line of detectors as shown in figure 7 (b) and (c).

The theorem that provides the mathematical basis for performing a CT reconstruction is the *Fourier Slice Theorem* . This theorem states that the one-dimensional Fourier transform of a parallel projection $P_\theta(t)$ is equivalent to a slice of the two-dimensional Fourier transform $F(u, v)$ of the original object $f(x, y)$ as shown in figure 8. More formally,

> The Fourier transform of a parallel projection of an image $f(x, y)$ taken at angle θ gives a slice of $F(u, v)$, the two-dimensional Fourier transform of $f(x, y)$, subtending an angle θ with the u-axis.

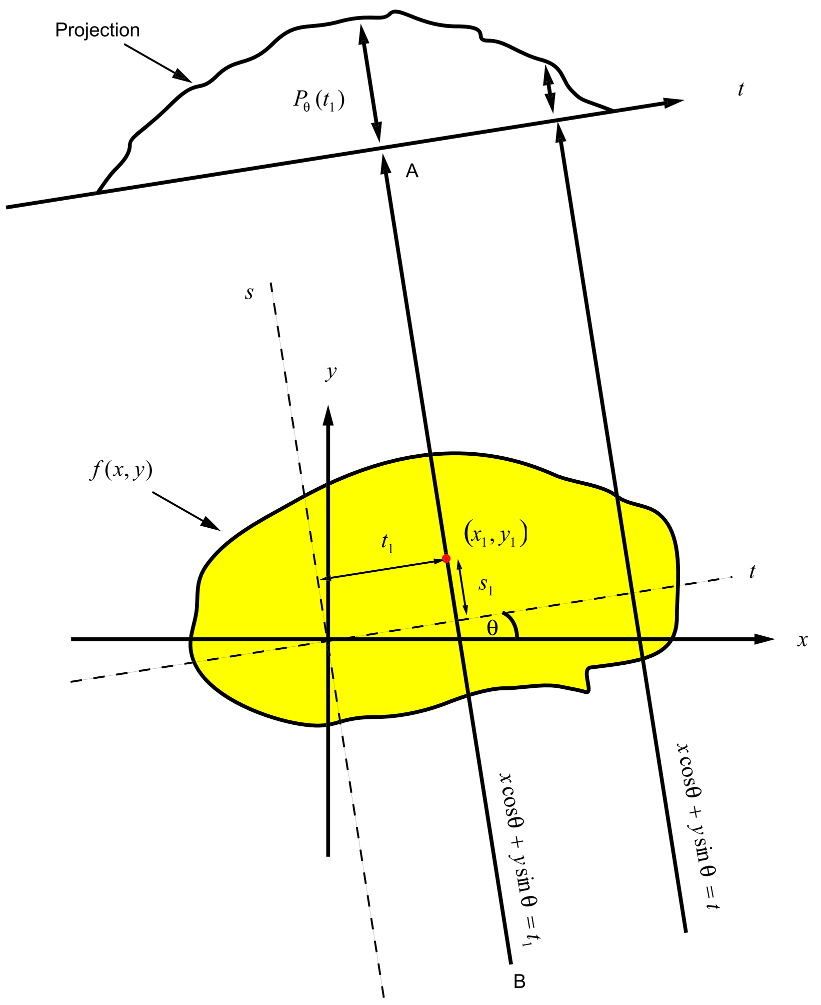

Figure 6: For an angle θ the projection $P_\theta(t_1)$ of an object $f(x,y)$ is shown

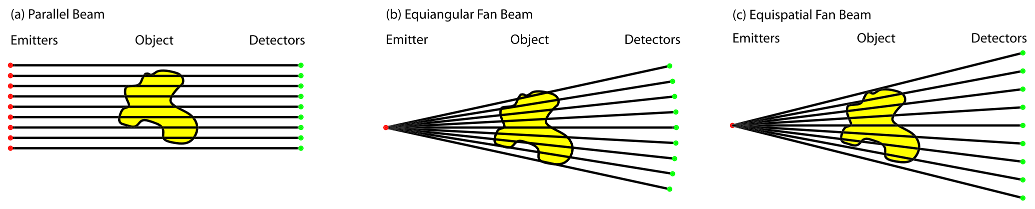

Figure 7: Different CT configurations

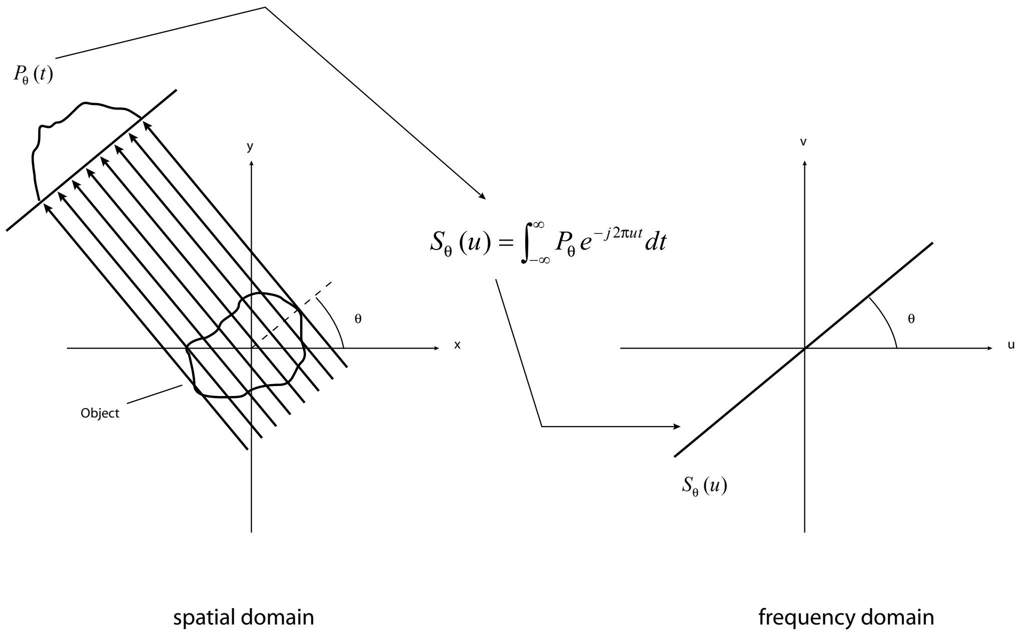

Figure 8: The Fourier Slice Theorem

Interested readers may wish to refer to [23] for a full derivation of this theorem. The implication of this theorem is that from a number of projections $P_\theta(t)$ we can estimate $F(u,v)$, and by performing an inverse Fourier transform on $F(u,v)$ we can estimate $f(x,y)$. $F(u,v)$ is defined as the two-dimensional Fourier transform of the original object.

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}\,\mathrm{d}x\,\mathrm{d}y \tag{19}$$

We also define $S_\theta(u)$ as the Fourier transform of $P_\theta(t)$.

$$S_\theta(u) = \int_{-\infty}^{\infty} P_\theta(t)e^{-j2\pi ut}\,\mathrm{d}t \tag{20}$$

Now according to the Fourier Slice Theorem we can estimate slices of $F(u,v)$ from a set of projections $S_\theta(w)$ taken at angles $\theta_1, \theta_2, \ldots, \theta_k$. Then, to obtain an estimation of the original object $f(x,y)$ it is necessary to perform an inverse Fourier transform

$$f(x,y) = \int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}\,\mathrm{d}u\,\mathrm{d}v \tag{21}$$

In practice, only a finite number of projections are taken. This means that the function $F(u,v)$ is only estimated from a finite number of radial lines emanating from the origin of the (u,v) Fourier coordinate system shown in figure 9.

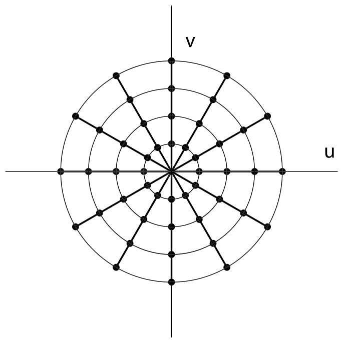

Figure 9: Estimate of the Fourier transform of the object $f(x,y)$ in the Fourier Domain, formed by the radial Fourier Transforms of a set of projections

Performing an inverse Fourier transform on such data would produce artifacts in the estimation of $f(x, y)$ since data is missing between the radial lines. In order to deal with this, the estimation of $F(u, v)$ needs to be filtered in order to interpolate data from existing projections. For the parallel projection configuration, this is as simple as taking a projection and using it to estimate a wedge of $F(u, v)$.

In practice, the estimation or reconstruction of $f(x, y)$ is determined by adding the two-dimensional inverse transform of each filtered projection over an image. This process is commonly called *filtered backprojection*.

The physical operation of a basic CT machine involves rotating the array of emitters and detectors around a patient and measuring x-ray projections at various angles. From this projection data, a two-dimensional slice of the internal structure of the patient is generated. The array is then moved along the axis of rotation and another slice is scanned. In this way a stack of two-dimensional slices of the patient are constructed which are then combined to create a three-dimensional volume.

The following algorithm summarises the CT reconstruction of a two-dimensional slice.

- For each angle of the N angles between 0 and $180°$
  1. Measure the projection $P_\theta(t)$
  2. Fourier Transform $P_\theta(t)$ to determine $S_\theta(u)$
  3. Filter $S_\theta(u)$
  4. Sum the inverse Fourier Transforms of the filter projections over the image plane

The first Computed Tomography scanner was constructed by Godfrey Hounsfield [19]. The significance of his invention showed that it was possible to generate highly accurate cross-sectional images even though projection data did not completely match the theoretical models underpinning the reconstruction algorithms.

Computed Tomography is the most commonly used method for generating three-dimensional data for medical purposes. CT machines are expensive and are usually only available in large South African hospitals. They also subject the patient to a significant amount of radiation since multiple x-rays must be emitted at various angles for each slice.

Chapter 3

Segmentation

This chapter describes the algorithms that we use to identify bullets in X-Ray images. Firstly, we examine the physical properties of bullets and the effects these properties cause when creating X-Ray images containing bullets. We then describe our model of a bullet in an X-Ray image. Finally we describe the algorithm that uses this model to identify the bullet within an image.

3.1 Related Work

A large amount of research has been directed towards segmenting various structures in x-ray images. Many approaches are used, depending on the structure that is being segmented.

Edge-tracking algorithms are some of the oldest segmentation techniques in use. However, it is difficult to identify an object from edge data since edges may be split up or “lost” by an algorithm amonst a group of edges. One of the more successful way of using edges to identify features in an X-ray is to search for edges relative to a spatial location. Thus, if a certain part of the body can be identified using some heuristic, the edges belonging to another part of the body can be recognised as important data.

Research has been performed in developing speech recognition algorithms that analyse the movement of the vocal tract from X-ray information. Thimm [46] uses a heuristic combined with a knowledge-based approach to segment and track the lips, teeth, palate and upper throat of a subject. Thimm first identifies the position of the teeth by analysing the x-ray histograms. From this information, the relative size of the head can be estimated. The rest of the structures are found by predicting their location with a template and correlating this with the edges produced by a Canny Edge Detector [9].

Brown et al. [7] also use a Canny Edge Detector and a knowledge-based approach to segment out the lungs of a patient. Long et al. [27] extract edges from x-rays and identify the jaw-bone as a reference point for finding the cervical vertebrae of the spine. Shimizu et al. also develop a method to segment out the lungs of a patient, based on an implementation of *Active Contour Models* (ACM) [24].

The knowledge-based approaches that we have described depend on first segmenting a structure that is highly visible and then searching for the desired structure based on its expected position relative to the first structure. This approach is not suited to searching for bullets since it is not possible to predict their position within the human body.

If the approximate shape of the structure that is being searched for is known, an *Active Shape Model* [21] (ASM) may be a good segmentation tool to use. Nopola et al. [34] use Active Shape Models to segment finger bones and Active Contour Models to segment wrist bones out of an x-ray image. This information is used to estimate the *bone age* of a patient which is useful for diagnosing conditions such as arthritis and diabetes. Lotjonen et al. [28] also use ASM's to segment bones out of x-rays, but use this information to provide a 3D reconstruction of the skeletal structure of a patient.

ACMs depend on strong edges to drive the model. This dependancy can be problematic since the edges of the region that is being searched for may be weaker than edges representing more spurious regions. In these cases the model will be attracted to the strong edges yielding an incorrect segmentation.

ACMs also require *a priori* information to position and initialise the model. Human intervention is often needed to initially position the snake. If a heuristic algorithm is used to position the snake, the actual usefulness of the snake is debatable since the heuristic could be extended to perform a full segmentation

ASMs provide an improvement on the basic ACM by searching for edges that fit a certain configuration. This version would seem better suited for a bullet searching algorithm. However, bullets may lose their shape when impacting with bodily structures or disintegrate into separate fragments.

Texture can also be used to segment structures within images. Action Region Models [22] extend ACMs to drive the model by the contents or texture of a region. Ivins approach to texture segmentation appears to be highly dependent on the snake parameters configured for each particular case.

Additionally, solutions based on Active Contour Models typically need to iterate towards a solution. This affects the speed of the operation, especially if accuracy is required.

Mammography, the procedure of analysing X-rays of the female breast to detect breast cancer, frequently makes use of texture analysis. Gupta et al. [16] use texture analysis to identify lesions in breast tissue.

Two approaches have been taken in texture analysis [35]. The first approach involves filtering an image to produce frequency sub-bands that represent the response of the filter to different textures. These response are then used to segment out the textured regions that produced the responses. The second involves generating statistical models for textures and using them to identify regions of texture.

Highnam et al. [17] have developed the h_{int} representation of a mammogram, that removes the noise caused by imaging conditions and associates with each pixel within an image, a thickness value indicating the “thickness” of the tissue at that point. This representation simplifies the process of segmentation since the removal of noise ensures that the features within a mammogram are better defined. The relative density of the breast tissue at each pixel can also be classified, which is useful for detecting abnormalities.

The h_{int} representation is extended in [18, 50] to account for blurring in the process of capturing a mammogram. This results in easier detection of breast calcifications since their sharp definition in mammograms is lost due to a combination of blurring of their small size.

The texture exhibited by bullets in X-rays tends be very uniform. While a texture analysis algorithm may be able to easily detect these homogenous regions, it would be excessive to use texture analysis algorithms when a simpler algorithm requiring less computing power could be used.

3.2 Analysis of the properties of a bullet in an X-Ray image

A useful approach to developing a segmentation algorithm is to understand how an image of an object is formed. In this subsection we will describe the physical properties of bullets and how these lead to the visual properties exhibited by bullets in x-ray images.

3.2.1 The Physical Properties of Bullets

Bullet are mostly composed of lead. Lead is one of the heaviest metal elements, with an atomic number of 82 and an atomic weight of 207.2 (atomic mass units) [49]. This mass is due to the 82 protons that make up the nucleus of a lead atom.

Because lead atoms are so dense, it is difficult for X-Rays to pass through materials composed of lead. Most of the X-Rays will be deflected. This is why lead is used to shield the operators of X-Ray machines from the effects of prolonged exposure to X-Ray radiation.

3.2.2 Properties of Bullets in X-Ray Images

As described in the previous section, the material used to construct bullets will deflect X-Ray particles. Therefore, a bullet will occupy a range of bright intensities within an X-Ray image since the area it occupies will not have been exposed to many X-Ray particles. This area will also be fairly uniform in intensity, again due to the lack of exposure.

These properties are useful in distinguishing between bullets and normal bodily structures within a Medical X-ray image. The human body does not contain any material as dense and heavy as lead. Human flesh and organs are largely composed of water, a very light molecule that easily allows X-Rays to pass through. Bone is a denser substance, but X-Rays can still penetrate it.

Therefore, bodily structures will occupy more varied ranges of intensity compared to a bullet. Since bullets occupy a very bright, uniform range of intensities, this property can be used as the basis for modelling a bullet within an image.

As can be seen in Figure 11 the bullet creates a very distinct plateau in the line profile taken across Figure 10. The plateau has steep side gradients and little variance in intensity across the plateau.

3.3 Model

Now that we have identified the properties of bullets in X-rays that distinguish them from other structures, we can model their image properties. Using the model, we can drive a search algorithm to find structures within X-ray images that correspond to the model. From our analysis of the line profiles of an image, we decided to use the distinctive plateau created by bullets as our model. We define our model of the plateau using Figure 12

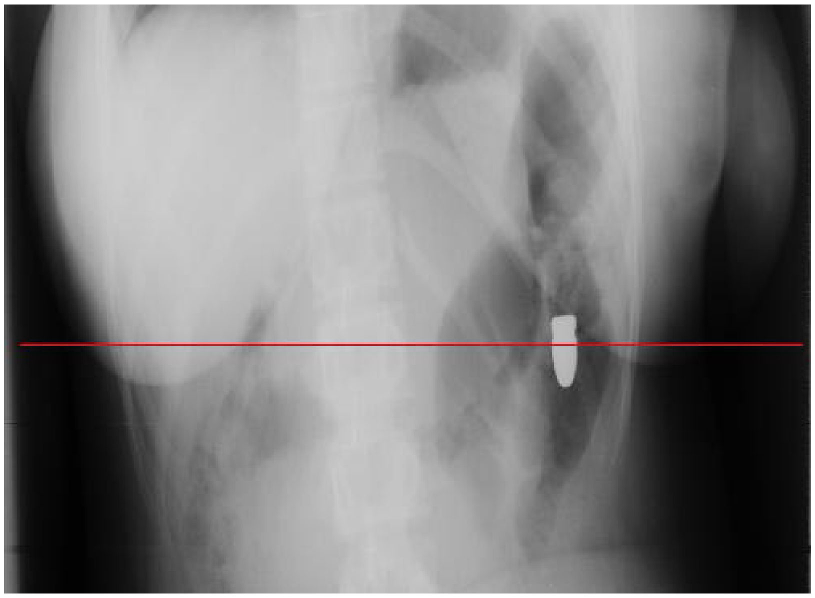

Figure 10: Original Image with Line Profile

1. The sides of the plateau have steep gradients and these gradients are similar. i.e. for some g_{min}:

$$\frac{f(i_{start}) - f(i_{end})}{i_{start} - i_{end}} > g_{min} \qquad \text{and} \tag{22}$$

$$\frac{f(i_{start}) - f(i_{end})}{i_{start} - i_{end}} \approx -\frac{f(d_{start}) - f(d_{end})}{d_{start} - d_{end}} \tag{23}$$

2. The increase in height from the bottom to the top of the plateau is significant. i.e. for some h_{min}:

$$f(i_{end}) - f(i_{start}) > h_{min} \qquad \text{and} \qquad f(d_{start}) - f(d_{end}) > h_{min} \tag{24}$$

3. We fit a linear regression to the values at the top of the plateau. The line XY represents this in Figure 12. Since the top of the plateau represents the area occupied by the bullet, this area should be fairly uniform. Thus, the average residual $r_{average}$ of the linear regression plot should be small, below some value r_{max}:

$$r_{average} = \frac{1}{n}\sum_{i=1}^{n} |r_i| < r_{max} \tag{25}$$

where r_i is a residual of the linear regression.

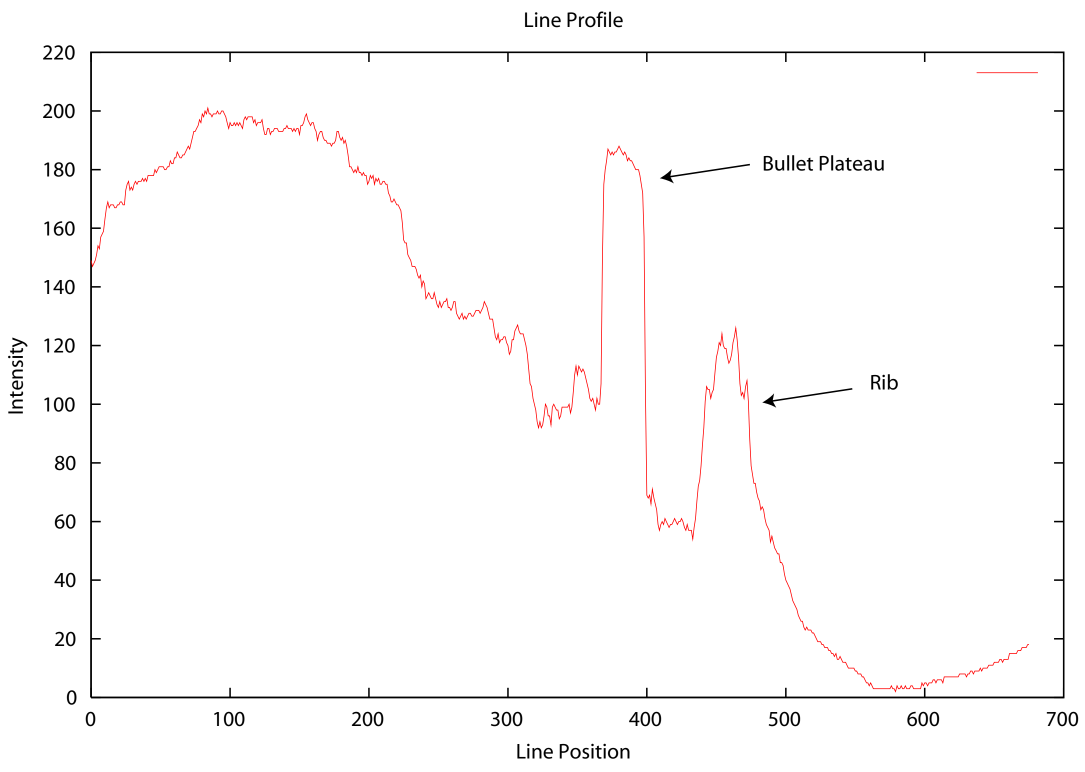

Figure 11: Line Profile of Figure 10. The bullet plateau can clearly be seen between Line Position 300 and 400

4. The angle that XY makes with the x-axis should not be too steep since the plateau of a bullet should normally be fairly flat. i.e. for some θ_{max}:

$$\theta_p < \theta_{max} \tag{26}$$

5. The width of the plateau should be wider than a certain value w_{min}:

$$d_{start} - i_{end} > w_{min} \tag{27}$$

These parameters need to be configured. They will need to be based on the typical profile exhibited for a particular X-ray machine configuration. X-ray machines are usually configured using *phantoms*. Phantoms are physical dummies that approximate bodily structures that the machines will be X-raying. By inserting a lead object into a phantom, the typical profile exhibited by lead for that particular machine can be established and used to configure the algorithm.

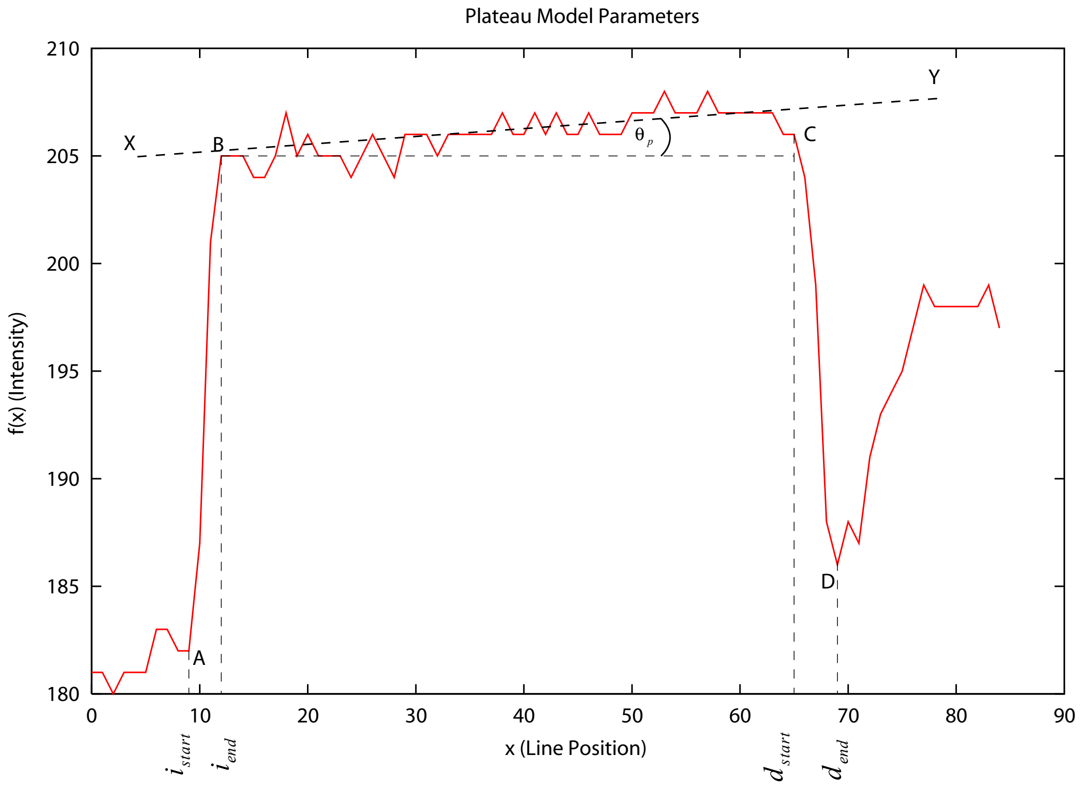

Figure 12: Visual Representation of some of the important features of our plateau model. $A = (i_{start}, f(i_{start}))$ is the start of the plateau's increasing gradient side $B = (i_{end}, f(i_{end}))$ is the end of the plateau's increasing gradient side $C = (d_{start}, f(d_{start}))$ is the start of the plateau's decreasing gradient side $D = (d_{end}, f(d_{end}))$ is the end of the plateau's decreasing gradient side. We also fit a *linear regression* to the top of the plateau. This is represented by the line XY. We measure the angle θ_p that XY makes with the x-axis

Since lead deflects so many X-rays, the profile will not change significantly for different types of lead objects. In fact the profile can be used to detect other types of heavy metal. Therefore, the algorithm will only need to be configured once during the setting up of an X-ray machine. In our Evaluation chapter, we describe the parameter configuration we use for testing our algorithm on X-rays.

3.4 Segmentation Algorithm

We now describe our algorithm for segmenting bullets out of x-rays. The aim of this algorithm is to identify regions of the image that may contain a bullet using the model described in the previous section. We first list the main elements of the algorithm and describe each section of it separately.

1. Examine the horizontal line profiles of each row in the input x-ray image to find line segments that fit the model described in section 3.3. Create an image of these line segments called h_{region}.
2. Examine the vertical line profiles of each column in the input x-ray image to find line segments that fit the model described in section 3.3. Create an image of these line segments called v_{region}.
3. Generate a image i_{region} of regions that correspond to intersection of h_{region} and v_{region}.
   (a) Perform disc erosion on i_{region} to produce i_{eroded}.
   (b) Perform disc dilation on i_{eroded} to produce $i_{dilated}$.
4. Backtrack from each region to identify the horizontal and vertical line segments that first created the region.
5. Use the line segments to identify the range of intensities that the plateau occupies. Threshold this range of intensities.

Algorithm 1 Algorithm to find matching increasing and decreasing gradient pairs

```
1: $P = \emptyset$
2: $k = 0$
3: $i_{start} = i_{end} = d_{start} = d_{end} = -1$
4: while $k < n - 1$ do
5:     Find a range of $x$'s $\{x_j | k \leq l \leq j \leq u < n - 1\}$ with some lower bound $l$ and upper bound
       $u$ such that
       $f(x_{j+1}) > f(x_j) \forall j$ (the range is increasing) or
       $f(x_{j+1}) < f(x_j) \forall j$ (the range is decreasing).
6:     if $f(x_{j+1}) > f(x_j) \forall j$ then
7:         if $f(x_u) - f(x_l) > h_{min}$ and $\frac{f(x_u) - f(x_l)}{x_u - x_l} > g_{min}$ then
8:             $i_{start} = x_l$
9:             $i_{end} = x_u$
10:        end if
11:    else
12:        if $f(x_l) - f(x_u) > h_{min}$ and $\frac{f(x_l) - f(x_u)}{x_l - x_u} > g_{min}$ then
13:            $d_{start} = x_l$
14:            $d_{end} = x_u$
15:            if $i_{start} \neq -1$ and $i_{end} \neq -1$ then
16:                $P = P \cup \{\{i_{start}, i_{end}, d_{start}, d_{end}\}\}$
17:                $i_{start} = i_{end} = d_{start} = d_{end} = -1$
18:            end if
19:        end if
20:    end if
21:    $k = x_u$
22: end while
```

Algorithm 2 This section of the main algorithm identifies horizontal and vertical line segments that match the plateau model

```
1:  Input an X-Ray image $i_{Xray}$
2:  Create an image $h_{region}$ of the same size as $i_{Xray}$
3:  Create a set $H_{rows} = \emptyset$.
4:  for all rows in the x-ray image defined by their height coordinate $y$ do
5:      Identify the set of candidate plateaus $P$ for the row using Algorithm 1.
6:      for all $p \in P$ do
7:          Fit a linear regression to the values of $i_{Xray}$ between $i_{end}$ and $d_{start}$ on row $y$. From this
            fit we obtain the angle of the regression $\theta_p$ and the average of the fit's residuals $r_{average}$.
8:          if $\theta_p < \theta_{max}$ and $r_{average} < r_{max}$ and $d_{start}$ - $i_{end} > w_{min}$ then
9:              Fill in the pixels between $i_{start}$ and $d_{end}$ in the corresponding row at height $y$ of the
                $h_{region}$ image.
10:             Set $H_{rows} = H_{rows} \cup \{i_{start}, i_{end}, d_{start}, d_{end}, y\}$
11:         end if
12:     end for
13: end for
14: Create an image $v_{region}$ of the same size as $i_{Xray}$
15: Create a set $V_{columns} = \emptyset$.
16: for all columns in the x-ray image defined by their width coordinate $x$ do
17:     Identify the set of candidate plateaus $P$ for the column using Algorithm 1.
18:     for all $p \in P$ do
19:         Fit a linear regression to the values of $i_{Xray}$ between $i_{end}$ and $d_{start}$ on column $x$. From
            this fit we obtain the angle of the regression $\theta_p$ and the average of the fit's residuals
            $r_{average}$.
20:         if $\theta_p < \theta_{max}$ and $r_{average} < r_{max}$ and $d_{start}$ - $i_{end} > w_{min}$ then
21:             Fill in the pixels between $i_{start}$ and $d_{end}$ in the corresponding row at height $y$ of the
                $v_{region}$ image.
22:             Set $V_{columns} = V_{columns} \cup \{i_{start}, i_{end}, d_{start}, d_{end}, y\}$
23:         end if
24:     end for
25: end for
```

Section 1 and 2: Identifying horizontal and vertical line segments. An image is a two-dimensional structure. This algorithm attempts to identify two-dimensional plateaus within this image. However, the model that we have specified can only identify plateaus in one dimension. By taking intensity profiles across the image and finding sections of the profiles that match our model, we can identify lines in the image corresponding to a cross-section of a two-dimensional plateau. If our model fits the plateau well, we can expect a large number of these lines to be present in the same area. We can then take advantage of this spatial coherence to to estimate a region containing the two-dimensional plateau.

The core of the algorithm is therefore based upon examining the intensity profiles of each row and column in an image. The sections of these profiles that match our model are accepted as candidate plateau cross-sections. We store the candidate cross-sections for the horizontal and vertical cases in the h_{region} and v_{region} images respectively.

Firstly we describe the algorithm we use to identify a horizontal plateau cross-section candidate . We take a line profile across a row of the X-ray image. We define the line profile as function $f(x)$ of the row position x. We also define a candidate plateau as the set $\{i_{start}, i_{end}, d_{start}, d_{end}\}$ where i_{start} and i_{end} define the x coordinates of the start and end of the steeply increasing segment of the plateau. d_{start} and d_{end} similarly define the start and end coordinates of steeply decreasing section of the plateau.

The process of identifying a set of candidate plateaus P along a row of n pixels is described in Algorithm 1. This algorithm searches for a range of x where $f(x)$ is constantly increasing. If the height of this range is greater than h_{min} and the gradient is steeper than g_{min} this range is treated as the side of a plateau and the extremes of this range are labelled i_{start} and i_{end}. The algorithm then tries to find a range of x where $f(x)$ is constantly decreasing and which also satisfies the h_{min} and g_{min} requirements. The extremes of this range are labelled d_{start} and d_{end}. These ranges are used to define a candidate plateau $\{i_{start}, i_{end}, d_{start}, d_{end}\}$ which is added to the set P.

Algorithm 1 is used to identify candidate plateaus based on the g_{min} and h_{min} parameters. However, these candidate plateaus must also satisfy the other three parameters of our model.

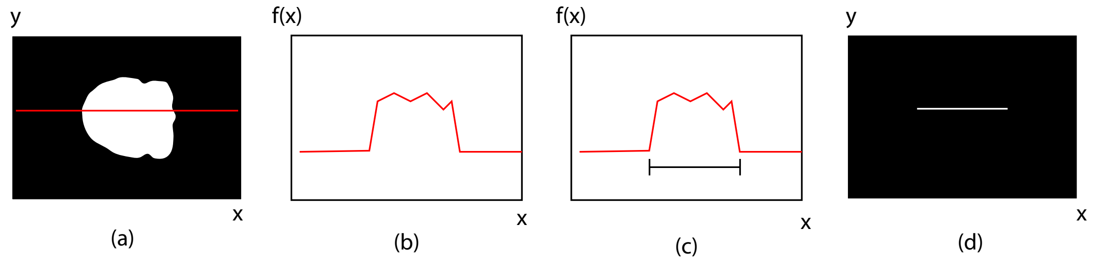

Figure 13: The process of identifying a candidate plateau and storing a corresponding line in the h_{region} image. (a) Line Profile across Image (b) Line Profile (c) Identification of Plateau in Line Profile (d) Storing corresponding line segment in h_{region}

We scan through the rows and then the columns of the x-ray image using algorithm 1 to identify candidate plateaus. For each candidate plateau we fit a linear regression to the top segment. From this regression we can extract variables for the flatness, θ_p and variability $r_{average}$, of the plateau. If the plateau is flat enough ($\theta_p < \theta_{max}$), its values are relatively stable ($r_{average} < r_{max}$) and is not too small ($d_{start} - i_{end} > w_{min}$) we accept the candidate. For the horizontal case, the accepted candidate's position is recorded as a line in the image h_{region} and a list of accepted candidates is maintained in H_{rows}. For the vertical case the image and list are v_{region} and $V_{columns}$ respectively. Figure 13 shows how a line is generated in h_{region} from a candidate plateau found along a line profile. This part of the main algorithm is listed in algorithm 2.

Figure 14 provides a graphical view of the segmentation process. Panels (b) and (c) show the h_{region} and v_{region} images that result from scanning in the horizontal and vertical directions.

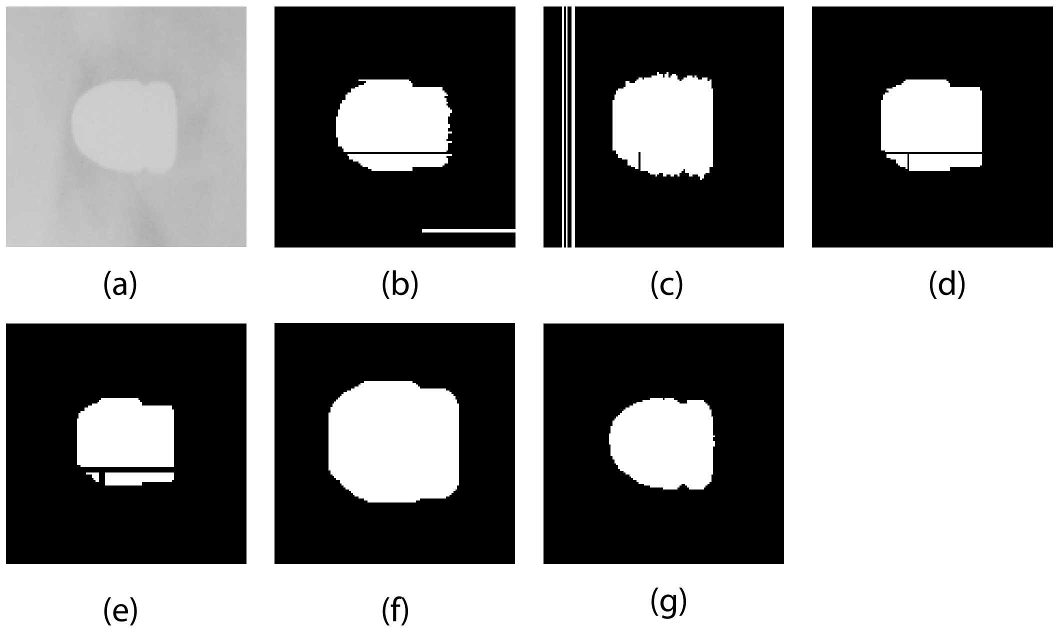

Figure 14: A step-by-step view of each stage of the segmentation process performed on a region of an x-ray containing a bullet (a) The Original Image (b) Horizontal Line Segments (c) Vertical Line Segments (d) Intersection of Horizontal and Vertical Line Segments (e) Eroded Image (f) Dilated Image (g) Final Segmentation

Algorithm 3 This section of the main algorithm describes how the intensities that are occupied by a two-dimensional plateau are identified

```
Create the output image $i_{segmented}$
Group the separate 4-connected regions of $i_{dilated}$ into set $R$.
**for all** regions $region \in R$ **do**
    Calculate the bounding box $B$ of $region$.
    Set $plateau\_max\_1 = 0$
    Set $plateau\_max\_2 = 0$
    Set $plateau\_min = 0$
    Set $count = 0$
    **for all** rows $r = \{i_{start}, i_{end}, d_{start}, d_{end}, y\} \in H_{rows}$ that intersect with $B$ **do**
        Set $increase\_halfpoint = \frac{i_{end} - i_{start}}{2}$
        Set $decrease\_halfpoint = \frac{i_{end} - i_{start}}{2}$
        $plateau\_min =$ **maximum**$(i_{Xray}(increase\_halfpoint, y), i_{Xray}(decrease\_halfpoint, y))$
        $plateau\_max\_1 =$ **maximum**$(i_{Xray}(v, y))$ where $i_{end} \leq v \leq d_{start}$
        $count = count + 1$
    **end for**
    **for all** columns $c = \{i_{start}, i_{end}, d_{start}, d_{end}, x\} \in V_{columns}$ that intersect with $B$ **do**
        Set $increase\_halfpoint = \frac{i_{end} - i_{start}}{2}$
        Set $decrease\_halfpoint = \frac{i_{end} - i_{start}}{2}$
        $plateau\_min =$ **maximum**$(i_{Xray}(x, increase\_halfpoint), i_{Xray}(x, decrease\_halfpoint))$
        $plateau\_max\_2 =$ **maximum**$(i_{Xray}(x, v))$ where $i_{end} \leq v \leq d_{start}$
        $count = count + 1$
    **end for**
    $plateau\_max =$ **maximum**$(plateau\_max\_1, plateau\_max\_2)$
    $plateau\_min = \frac{plateau\_min}{count}$
    **for all** pixels $b \in B$ such that $plateau\_min \leq i_{Xray}(b) \leq plateau\_max$ **do**
        $i_{segmented}(b) = 1$
    **end for**
**end for**
```

Section 3: Using Spatial Coherence to Identify Two-Dimensional Candidate Regions. At this stage of our algorithm, we have identified lines that fit our model in the horizontal and vertical direction. These lines may correspond to the cross-section of a two-dimensional plateau. In order to determine whether this is true, we take advantage of the fact that if a two-dimensional plateau is present the horizontal and vertical cross-sections of this plateau will overlap with one another. To determine the region of overlap, and therefore the approximate shape of the two-dimensional plateau, we intersect the h_{region} and v_{region} images with each other to produce the i_{region} image. This intersected region is shown in panel (d) of Figure 14.

The regions that are produced by this operation may be insignificant and only consist of two or three pixels. In order to remove these spurious regions we perform a *disc erosion* [43] operation with a radius of three pixels on the i_{region} image to produce i_{eroded}. An example of the erosion process is shown in panel (e) of Figure 14.

A sequence of plateau cross-section candidates in the h_{region} or v_{region} images may contain gaps where a row or column did not fit the model criteria. An example of this can be seen in Figure 14 (b). When the intersection operation between h_{region} and v_{region} occurs, these gaps are propagated to the region representing the plateau in the i_{region} image. The gaps are exaggerated when the disc erosion operation occurs. We therefore perform a *disc dilation* [43] operation to expand the region and fill these gaps. We use a disc radius of nine pixels to make sure these gaps are filled. This also results in the region being slightly larger than the actual area of the bullet as shown in panel (f) of Figure 14. The region represents the local area containing a strong plateau, formed by the intersection of one-dimensional plateau candidate lines.

Section 5 and 6: Backtracking to determine the range of intensities occupied by the plateau. By this stage of the algorithm, we have determined a number of two-dimensional regions that contain strong plateaus. We now need to determine the exact region occupied by each plateau. To do this we return to the one-dimensional plateau candidates that originally created these regions. We use the fact that these one-dimensional plateaus define a range of intensities occupied by the two-dimensional plateau. By examining each one-dimensional plateau we can estimate the range of intensities occupied by the corresponding two-dimensional plateau. Using this technique we can segment each plateau out of the corresponding region in $i_{dilated}$ with a simple threshold operation.

We now describe this section of the algorithm in more detail. Firstly, we examine the regions that exist in $i_{dilated}$ and associate these regions with the horizontal and vertical plateau candidates that contributed to their creation. This is accomplished by testing if the candidate plateaus intersect with

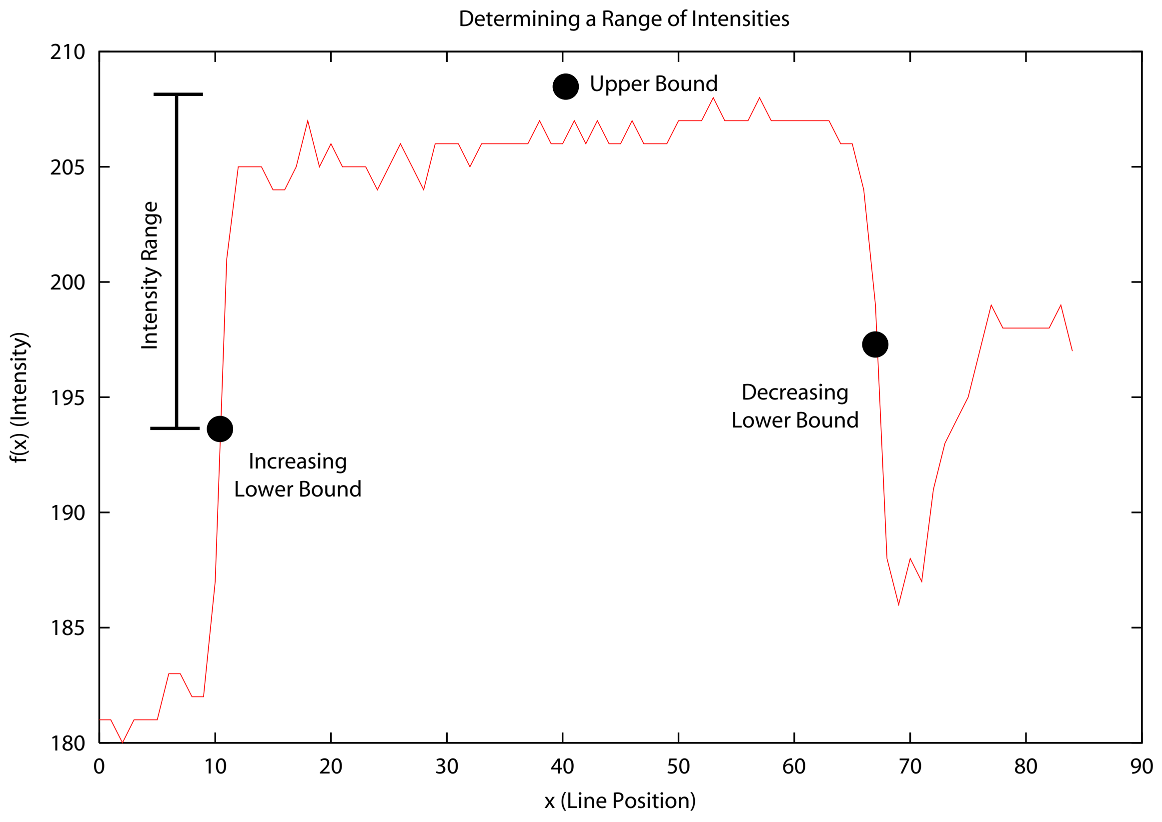

Figure 15: Determining the range of intensity for thresholding purposes. The lowest lower bound is chosen as the bottom of the intensity range. The maximum intensity value for the plateau, plus a "buffer" value is chosen as the top of the intensity range

the bounding box of each region. These plateau candidates are retrieved from the H_{row} and V_{column} sets that we determined in part 1 and 2 of our algorithm.

Since a plateau candidate represents a curve of intensities, we can examine the curve to determine the lower and upper intensity bound of the plateau. By examining the intensity curves of all the plateau candidates that contribute to a region, we determine the average lower and upper intensity bound of the plateau contained within a particular region. We then use thresholding to extract this range of intensities out of the bounding box of the region.

A graphical view of the thresholding intensity bound selection process is shown in Figure 15. The upper bound is simply set to the highest intensity value found amongst the plateaus. A small "buffering" value is added to the upper bound to ensure that all high intensities are included. This is because these high intensity values may have occurred in intensity profiles that were discarded by our algorithm for not meeting other parameters.

In practice, the lower intensity bound of a plateau is found by examining the intensity values of the halfway points on the steeply increasing and decreasing sections of the curve ie. $f(\frac{i_{end}-i_{start}}{2})$ and $f(\frac{d_{end}-d_{start}}{2})$ and selecting the lowest one. The reason for this is that the steep section of the curve represent strong edges and the halfway point these edges traditionally dilineate the boundaries of a region [29] in edge-detection algorithms. This final section of the main algorithm is described in algorithm 3.

3.5 Summary

In this chapter we briefly mention a number of techniques that are used to segment structures out of X-rays. A discussion on the physical structure of a bullet and the image properties of a bullet follows. We developed the model we used to drive our segmentation algorithm. Finally we described our algorithm to segment bullets from X-rays.

Chapter 4

Reconstruction

In this chapter we present the reconstruction algorithm that we use to estimate the three-dimensional shape and position of the three-dimensional object from its two-dimensional X-ray projections. Reconstruction algorithms can be divided into two different approaches, those that are based on computational geometry and those that are voxel-based. We examine the advantages and disadvantages of using these techniques.

Next, we describe our reconstruction algorithm. We provide an overview of the octree data structure which we use to represent our reconstructed object. The input data that the reconstruction algorithm operates on is detailed, as well as the constraints that we place upon this data to simplify the algorithm. Finally we describe the actual algorithm.

4.1 Related Work

Our reconstruction algorithm is based upon the concept of the *Visual Hull* first proposed by Baumgart [4], and placed upon a firm theoretical basis by Laurentini [2]. Visual Hull Reconstruction Algorithms have also been called *Volume Carving* algorithms since they perform intersection operations of silhouette projections. The volumetric nature of the data has resulted in two approaches in respect to Volume Carving: techniques based on voxel and geometric representations.

Voxel-based techniques are storage intensive if the volume representation is implemented as a three-dimensional array . Octrees address this problem by hierarchically representing voxel data to take advantage of *spatial coherence*. This involves grouping voxels within spatial areas and testing whether they all share the same value. If this is the case, all the voxels within a spatial area can be represented by the area and the shared intensity value, drastically saving on the amount of memory required to represent a voxel volume. Chien et al. [11] reconstruct an object from three orthogonal views to produce an octree representation of the final object. Veenstra et al. [47] perform an octree reconstruction from thirteen prescribed orthogonal viewpoints.

Potmesil [38] also presents an octree reconstruction based upon creating octree representations of silhouette projections and intersecting them to form a final octree representation. His approach allows for multiple arbitrary viewpoints and perspective projections. Szileski [45] improves upon this by maintaining a single octree that is refined by projections.

Voxel-based techniques suffer from resolution problems [8]. It is difficult to determine the voxel resolution required to accurately represent the object that is being reconstructed. Octree representations ameliorate this somewhat. However, voxel-based techniques provide fast reconstruction since intersection of silhouette projections simply involves testing whether a voxel is in both projections.

Martin and Aggarwal [30] reconstruct slices of a three-dimensional volume by intersecting parallelograms derived from projection cones. Petitjean[37] extends Laurentini's [2] work using visibility graphs to improve the algorithmic complexity for 2D and 3D reconstruction. Matusik et al. [31] perform 3D reconstruction by projecting the 3D visibility cones onto 2D planes, which are then intersected with the projections of other 3D cones.

Geometrically-based techniques are more complex than voxel-based methods and are difficult to implement robustly [8]. Silhouettes consisting of many segments will result in the generation of a large number of faces in the reconstructed model. Implementing reconstruction from 3D projections of silhouettes is also more challenging than the 2D case [2, 37]. Geometrically-based methods do not suffer from the resolution problems found in voxel-based methods.

4.2 Reconstruction Algorithm

Our reconstruction process is relatively simple since it involves the reconstruction of the two-dimensional slices of a three-dimensional volume. We do this because the two-dimensional case is not as as challenging to implement as the three-dimensional case. Chien et al. [11] take this approach when they perform a reconstruction from three orthogonal views as this can be reduced to intersecting two-dimensional slices together. Matusik et al. [31] project three-dimensional silhouettes onto two-dimensional planes so that the two-dimensional case can be used.

More general algorithms exist that allow a reconstruction to be performed from arbitrary viewpoints [38]. However, due to time constraints we implemented a simple reconstruction algorithm that has a number of constraints on the input data. Future Work may involve implementing a more general algorithm that does not have these constraints.

Firstly, we describe the octree volume representation that we will use to perform our reconstruction operation on. Secondly, we describe the input data and the constraints upon the input data and finally we describe the actual reconstruction process.

4.2.1 Octree Volume Representation

We use an octree to represent the volumetric data that we obtain from our reconstruction algorithm. An *octree* is an hierarchical data structures that recursively subdivides a three-dimensional space. It is the three-dimensional version of the two-dimensional quadtree [25, 3, 40].

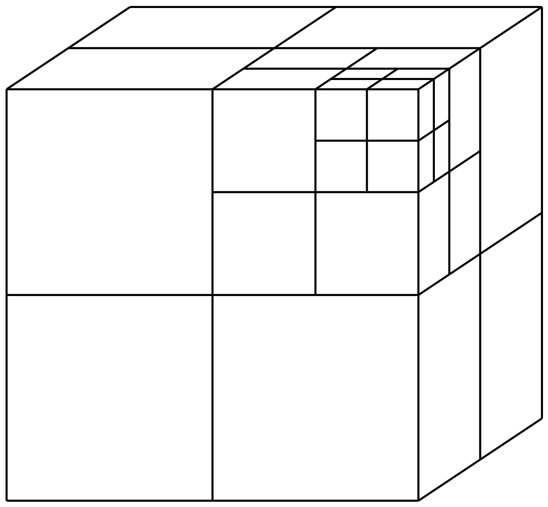

Figure 16: Subdivision of the top lower right octant of an octree

The most common form of octree consists of a tree of nodes, each describing a cube-shaped region of space and the contents of that space. Each node contains a descriptor, describing the contents of the cube, and eight child nodes. The descriptor can be set to WHITE, BLACK and MIXED. The descriptor is set to WHITE if the entire region in the cube is WHITE and BLACK if the entire region

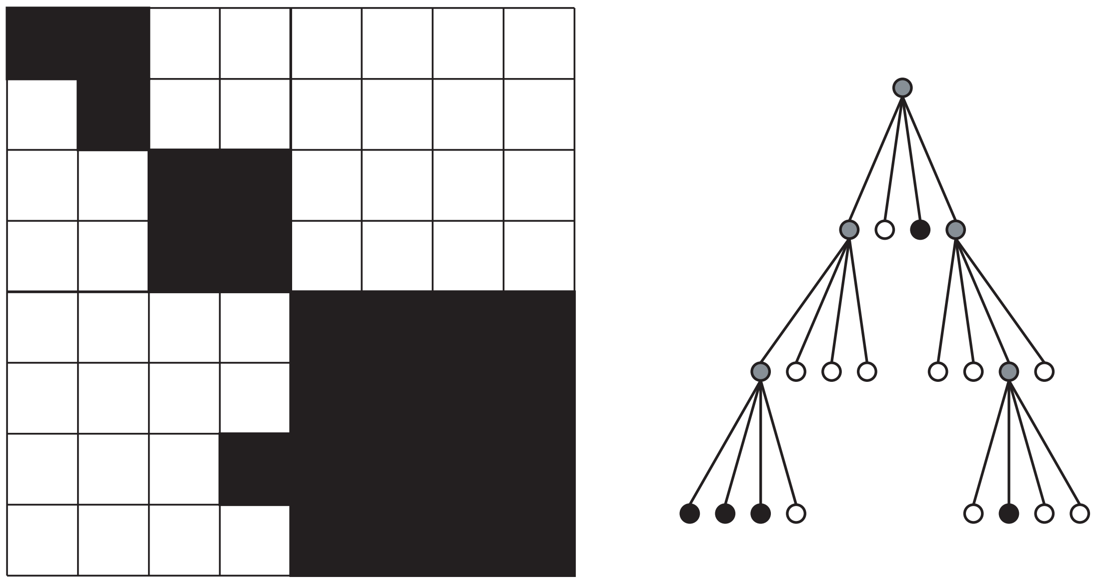

Figure 17: An image and the quadtree derived from it. White and black nodes represent space occupied by WHITE and BLACK markers respectively. Grey nodes represent space occupied by both WHITE and BLACK markers. The quadrants correspond to child nodes in the following order, northwest, northeast, southeast, southwest.

is BLACK. If the cube describes a region of space containing both WHITE and BLACK values, the descriptor is set to MIXED and the node is subdivided into its eight child nodes.

These child nodes subdivide the cube occupied by the parent node into eight cubes or octants as shown in Figure 16. This subdivision continues until each node's descriptor is either WHITE or BLACK, or a one unit resolution has been achieved. Due to this subdivision operation, the dimensions of the cube are usually powers of two, so that subdivision results in a cube or voxel that has a dimension of one unit. In Figure 17, we show how the tree structure of a *quadtree* is constructed from an image as it is somewhat easier to understand the process in two dimensions. Note that WHITE or BLACK nodes are leaf nodes while MIXED nodes are always parent nodes.

One of the most useful properties of an octree is that it heirarchically describes the contents of a three-dimensional space. It is therefore advantageous to use octrees in cases where the spatial data exhibits a high degree of *spatial coherence*. For example, if an octant with a width, height and depth of 16 only contains a BLACK region of space, the contents of this region can be stored in the octant's node without resorting to further subdivision. By taking advantage of the *spatial coherence* of this region of space the voxels in this region can be represented more concisely as (a) the region and (b) the intensity of the voxels within the region. This results in large savings in the memory cost of the volume representation when compared to a more naive implementation as a three-dimensional array.

Due to this hierarchical representation, relevant regions of space can quickly be found using a divide and conquer approach. Since an octree is a tree structure, insertion and deletion operations have an algorithmic complexity of $O\log(n)$ which are slightly more expensive than the corresponding $O(1)$ complexity operations on a three-dimensional array. This tradeoff in speed is well-worth the savings in memory.

4.2.2 Input to the Reconstruction Algorithm

The input to the reconstruction algorithm is a sequence of X-ray images and the associated projection data for each X-ray image. The X-rays are images of a bodily structure containing a bullet. In order to effect a good reconstruction these images should be taken at a variety of different angles. The projection data describes the path that an X-ray projection took from the X-ray source to the image.

Our algorithm accepts X-ray images in the gray-scale BMP file format. This can be easily changed to do deal with other formats. We define w and h to be the width and height of the image respectively. Each image needs to have an associated file describing the projection data for the image. This file contains a series of two-dimensional projections $\{P_1, P_2, \ldots, P_h\}$, each corresponding to the h rows of the image.

Each projection $P_i = \left\{l_1^i, l_2^i, \ldots, l_w^i\right\}$ describes the paths of a set of line integrals l_j^i which formed the row of w pixels within the X-ray image. In other words, each pixel (x, y) has an associated line integral $l_x^y \in P_y$ that describes the path that an X-ray took from the X-ray source to the point on the plane on which the X-ray image was projected. Each line integral $l_j^i = \left\{\vec{a}, \vec{b}\right\}$, consists of two vectors, $\vec{a}$, the gradient of the line and $\vec{b}$ the offset of the line. This representation is quite verbose compared to other representations such as projection matrices or cones of projection. However, it allows for greater flexibility in specifying different types of projections.

This projection data will be used to back-project bullet silhouettes from an X-ray image in order to reconstruct the bullet. Therefore, the coordinate system of the reconstructed data will correspond to that of the projections that create it.

It may be necessary to scale the projection data in order to suit the resolution of the voxel matrix. This can be performed as a pre-processing step.

As we have mentioned earlier, our reconstruction process reconstructs two-dimensional slices of the three-dimensional reconstruction volume. In order to simplify the reconstruction procedure to a two-dimensional case we place the following constraints on our input data:

1. An X-ray image is a projection of an object onto a two-dimensional plane in a three-dimensional space. The planes containing the X-rays must be parallel to the Z-axis.
2. The projections that form the X-ray image must be perpendicular to the Z-Axis. This means that the line integrals that form a single projection must all exist in a plane that is perpendicular to the Z-axis. In other words the projections must be two-dimensional.

These two constraints are shown in Figure 18. By introducing these constraints, the reconstruction algorithm can be reduced to the two-dimensional case. By requiring the X-ray image planes to be parallel to the Z-axis, we ensure that a row in one image shares the same projection space Z-coordinate with an equivalent row in another image. By requiring projections to be perpendicular to the Z-axis we ensure that when a row in an image is back-projected, it is only back-projected over one slice of the three-dimensional reconstruction volume.

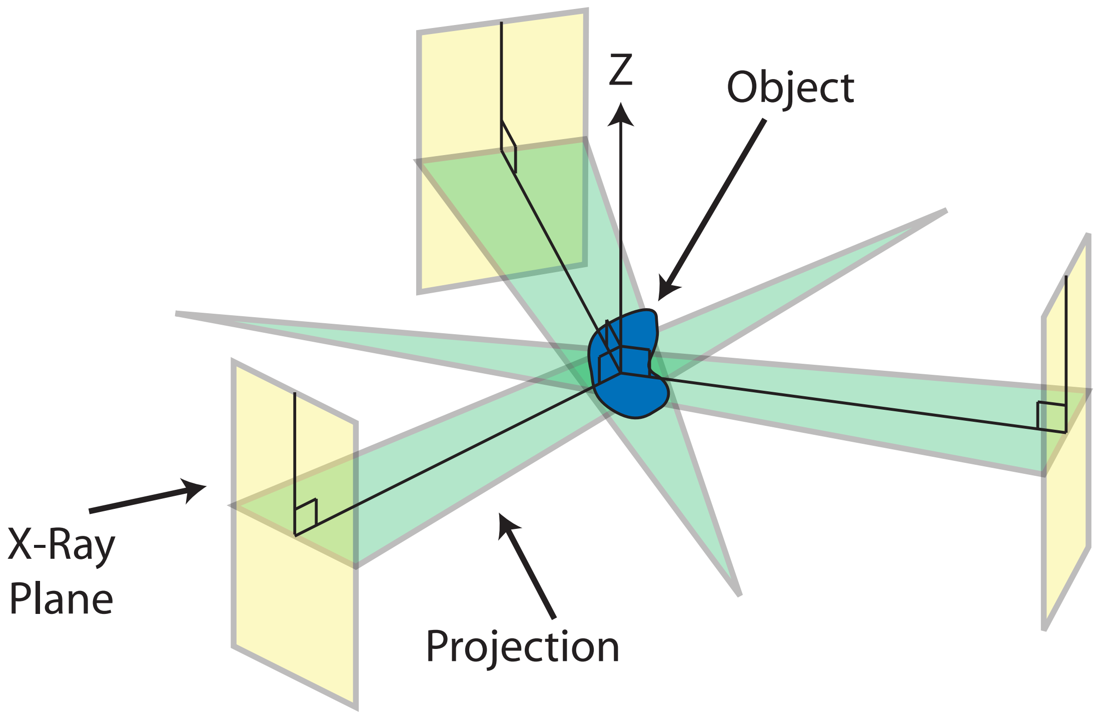

Figure 18: This diagram illustrates the constraints that X-rays used in the reconstruction process must adhere to (1) The two-dimensional planes onto which the X-ray images are projected must be parallel to the Z-axis (2) The projections that form the X-Ray image must be perpendicular to the Z-axis

The reconstruction algorithm is now reduced to a two-dimensional case because equivalent rows of pixels in the set of input images correspond to a single slice of the reconstruction volume.

The consequence of introducing these constraints is that our algorithm will not be able to deal with the data generated from an X-ray scanner that operates in a spiralling motion. This modality is the standard mode for CT [48]. It trades off some reconstruction accuracy for faster acquisition time. This reduction in the amount of time the patient needs to keep still reduces the temporal error associated with capturing the data. Our technique is therefore restricted to use with "stepping" modalities, where a slice of an object is scanned from a number of angles and the scanner is then stepped up to the next slice.

4.2.3 Reconstruction Process

The aim of this process is to generate a three-dimensional model of an object from its two-dimensional projections. We represent this three-dimensional *reconstruction volume* as an octree which uses a large range of integers to mark different regions of space. We use 0 to mark an empty region of space and positive integers to mark regions of space occupied by the object that we are reconstructing. The entire reconstruction volume is initially marked with 0: empty space.

By allowing the entire range of positive integers to be used as a mark, we reduce the effectiveness of the octree's hierarchical decomposition in describing the volume containing the object. This is because the nodes of the octree that describe sections of the volume containing the object will tend to be MIXED, leading to subdivision right down to the voxel level.

However, we expect most of the reconstruction volume to be occupied by empty space. Therefore, we still take advantage of the octree's hierarchical decomposition property over large sections of the reconstruction volume. By allowing the volume occupied by an object to be represented as the range of positive integers, we will be able to determine how many projections contributed to creating an object voxel. This will be used to determine whether a voxel should be retained or discarded in our final estimation of the reconstructed object. In this sense, the reconstruction volume is a three-dimensional *accumulation buffer*.

The process starts by reading in an X-ray image that contains a bullet and the projection data for the X-ray. The segmentation algorithm described in Chapter 3 is then applied to the image, producing a region that describes the area occupied by the bullet. This is the silhouette that will be back-projected over the reconstruction volume.

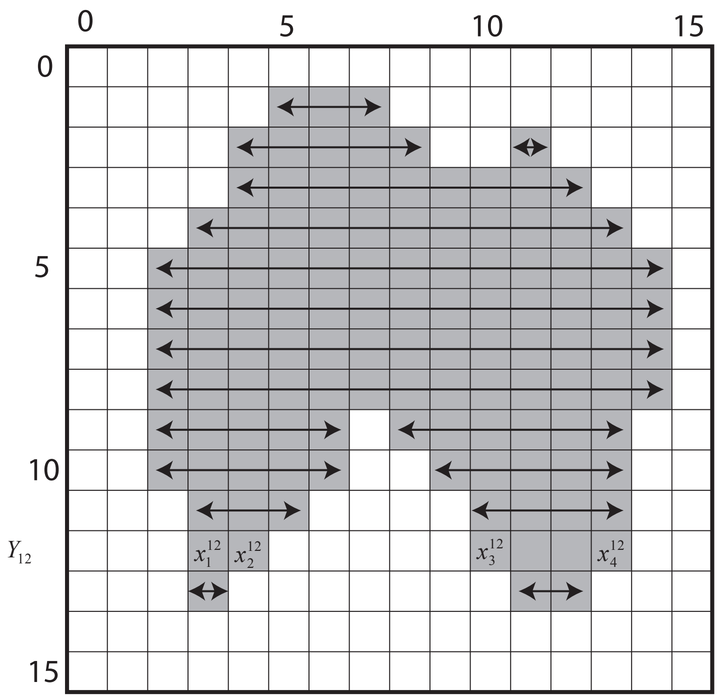

Figure 19: A series of pixel spans that represent a region of an image

We back-project one row of the silhouette at a time since a row corresponds to a slice of the reconstruction volume. To do this, we represent the silhouette as an table of *pixel spans* [32]. This data structure represents an image region R_k as a series of rows, each containing a number of pixel *spans*. Each span describes a horizontal set of contiguous pixels that contribute to the region as shown in Figure 19.

The pixel span table is constructed from a region's contour. See [32] for an explanation of this process. The contour of a region can be extracted using the algorithm described in [15]. P_k, the pixel span table representation of R_k is defined as follows

$$P_k = \left\{ y^k_{min}, y^k_{max}, Y^k_1, Y^k_2, \ldots, Y^k_{n_k} \right\} \tag{28}$$

y^k_{min} and y^k_{max} are the lowest and highest points on the region's contour respectively. The Y^k_i

symbols refer to the separate rows of P_k. There are $y^k_{max} - y^k_{min} + 1 = n_k$ rows in P_k. Each row Y^k_i contains a number of pixel spans and is defined as follows

$$Y^k_i = \left\{ r_{ik}, x^{ik}_1, x^{ik}_2, \ldots, x^{ik}_{r_{ik}} \right\} \tag{29}$$

The x^{ik}_j are points on the contour of R_k that are used to define the pixel spans. r_{ik} is used to store the number of points in the span. r_{ik} is always even since it takes two points to define the start and end of a span. In Figure 19, we show $Y_{12} = \left\{ r_{12} = 4, x^{12}_1 = 3, x^{12}_2 = 4, x^{12}_3 = 10, x^{12}_4 = 13 \right\}$.

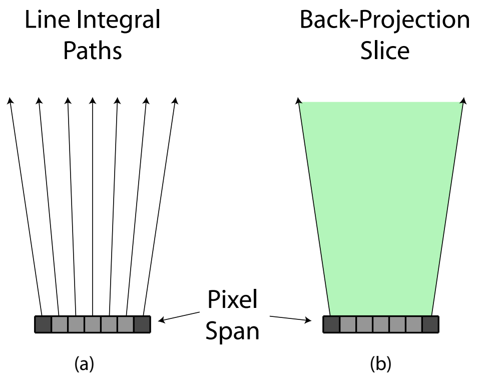

Figure 20: Creating a back-projection slice from a pixel span. (a) Each pixel in the pixel span has a line integral associated with it. (b) We define the back-projection slice as the space between the line integrals of the starting and ending pixels.

Now that the silhouette is represented as a pixel span table, we iterate through the rows of the table, back-projecting pixel spans. Due to the constraints that we have specified, each row of the image corresponds to a slice of the reconstruction volume. Therefore, the y coordinate describing the row of an image can be used to determine which slice of the reconstruction volume that row is back-projected over.

Firstly, we create a *back-projection slice* from the span that we are back-projecting. As described in section 4.2.2, each pixel (x, y) in the row has a line integral l^y_x associated with it as shown in Figure 20 (a). We define a back-projection slice B as the space between the line integrals $l^y_{x_i}, l^y_{x_{i+1}}$ associated with the pixels x_i, x_{i+1} at the start and end of the span. This is shown in Figure 20 (b).

Once the back-projection slice has been determined, it is intersected with the corresponding slice of the reconstruction volume as shown in Figure 21. However, the back-projection slice must first be

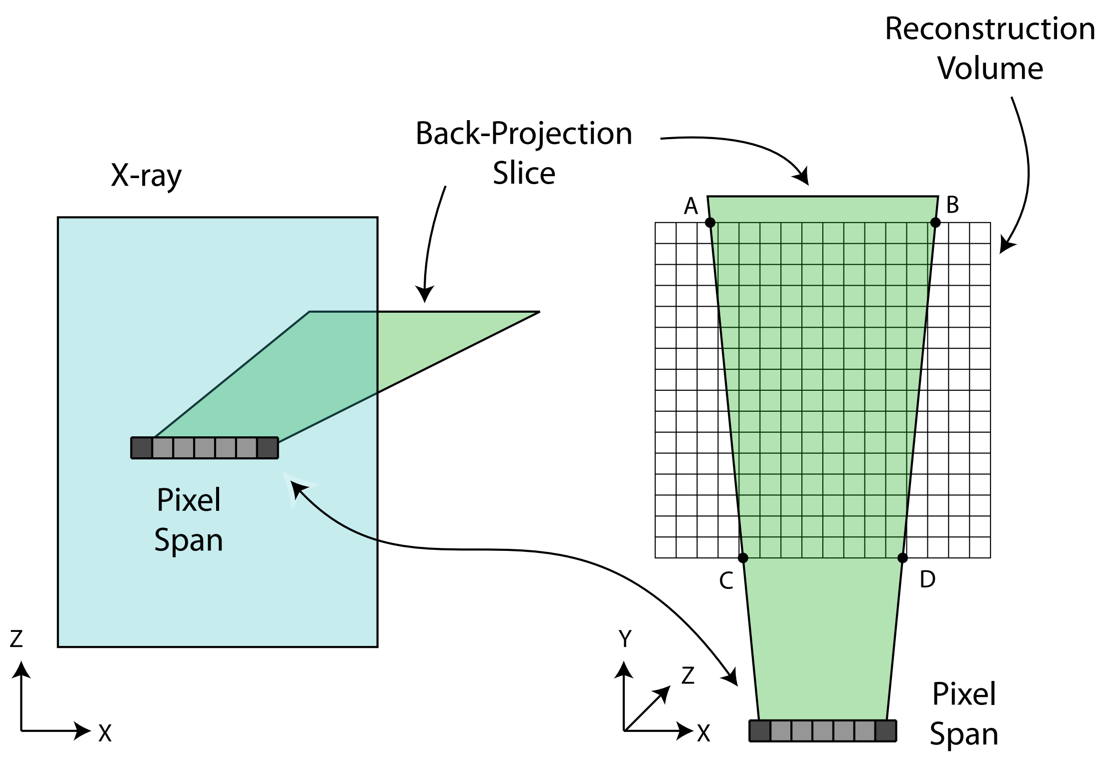

Figure 21: Intersecting the *back-projection slice* with the reconstruction volume

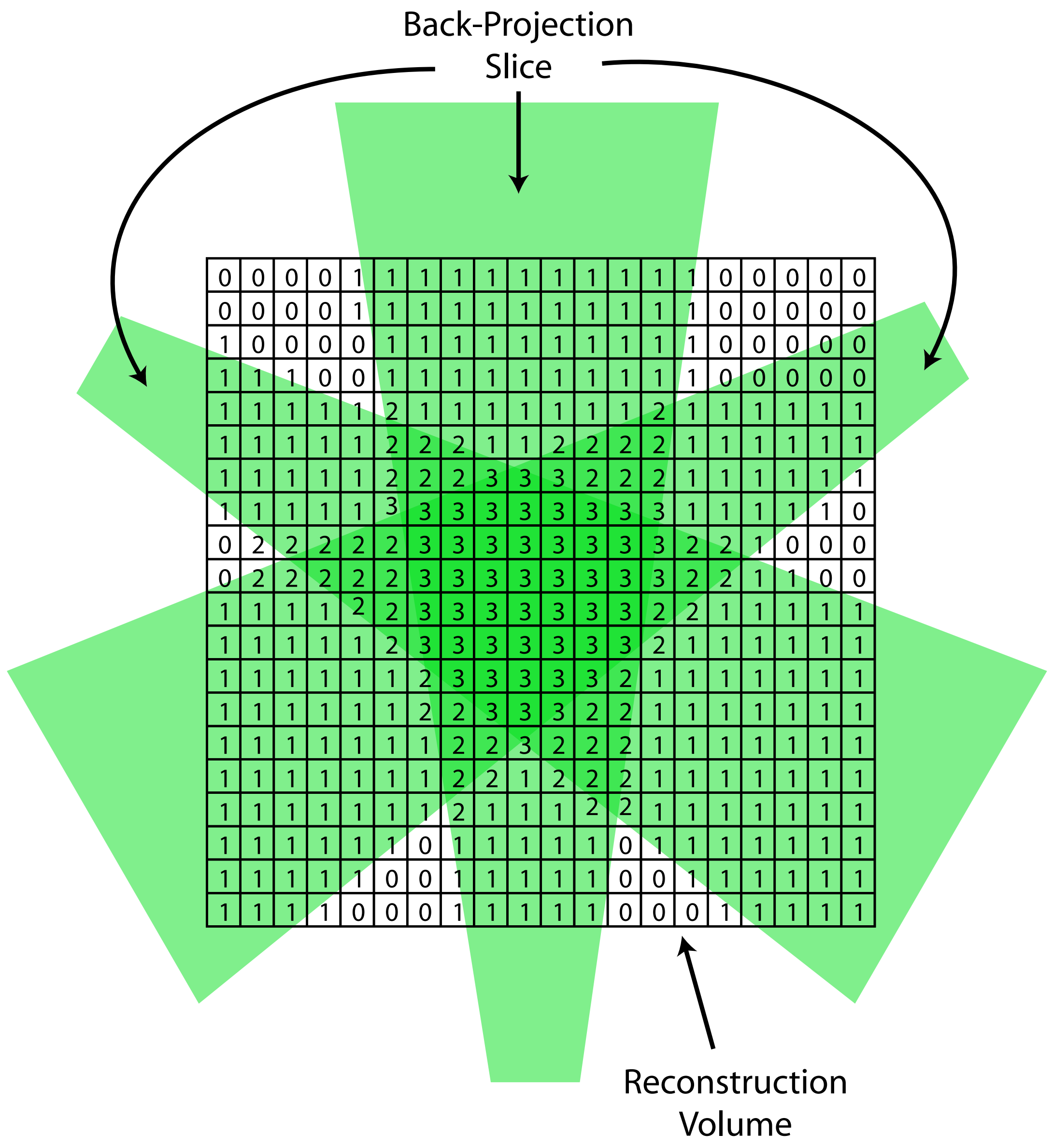

Figure 22: The result of intersecting three back-projection slices on the reconstruction slice. The labelling process results in the voxels being marked with the number of images that were back-projected through the voxel. Note that the area labelled with 3 is the visual hull for the three projections.

discretised. To do this, we clip the back-projection slice with the outer bounds of the reconstruction slice to produce a polygon. The polygon $ABCD$ in Figure 21 is an example of this.

This polygon is then converted to a two-dimensional voxel span table representation which is used to indicate which areas of the voxel slice are occupied by the polygon. This structure is exactly the same as a pixel span table. One operates over an two-dimensional image, the other over a two-dimensional slice of voxels. We term it a voxel span table to distinguish the type of data it is iterating over.

The intersection operation is then easy to perform. The reconstruction volume is a three-dimensional accumulator and the intersection merely involves performing an accumulation operation over the voxels in the table. We iterate through the voxel spans in the table. For each voxel the a span, we add 1 to its mark value. Once we have intersected all the voxel spans, we can move onto another pixel span from the X-ray image, and once all the pixel spans have been back-projected we move on to another X-ray image.

By incrementing voxels values we determine how many images contributed to the voxel. If the input to the reconstruction algorithm was 12 X-ray images, a voxel could have 13 different values, 0 if it was empty, or 1–12 depending on how many images managed to back-project over that voxel. We can use this information to determine whether the voxel is in the visual hull or not. If a voxel has a value of 12, it is in the visual hull of the reconstructed object since all 12 images contributed to it. An example of this is shown in Figure 22.

The final step of the reconstruction algorithm involves iterating over the reconstruction volume and thresholding voxels out. For example, for an input of 12 X-ray images, we would scan through the reconstruction volume, searching for voxels with a value of 12. We then set these voxels to 1 and any other voxels to 0.

Algorithm 4 describes the main section of the reconstruction algorithm. The back-projection section of the algorithm is described in Algorithm 5.

Algorithm 4 Main Reconstruction Algorithm

```
1: Initialise an octree $O$ of width, height and depth $d$
2: $nimages \leftarrow 0$
3: **while** Input X-ray images remain **do** {Determine an image's contribution to the Visual Hull}
4:    $nimages \leftarrow nimages + 1$
5:    Input an X-ray image $I$ of width $w$ and height $h$.
6:    Input the projection data $P = \{P_1, P_2, \ldots, P_h\}$ associated with image $I$
7:    Apply segmentation algorithm from Chapter 3 to $X - Ray$ to produce a set of regions $R = \{R_1, R_2, \ldots, R_n\}$, containing bullets
8:    **for all** $R_j \in R$ **do**
9:       Extract the contour $C_j$ of region $R_j$
10:      Construct a pixel span table $\mathrm{PST}_j = \left\{y^j_{max}, y^j_{min}, Y^j_1, Y^j_2, \ldots, Y^j_{n_j}\right\}$ of region $R_j$ from its contour $C_j$
11:      **for all** rows $Y^j_k \in \mathrm{PST}_j$ **do**
12:         $y \leftarrow k + y^j_{min} - 1$
13:         $row \leftarrow y$
14:         **for all** pixel span pairs $x^{jk}_i, x^{jk}_{i+1} \in Y^j_k$ **do**
15:            $x_s \leftarrow x^{jk}_i$
16:            $x_e \leftarrow x^{jk}_{i+1}$
17:            Create back-projection slice $B = \{row, l^y_{x_s}, l^y_{x_e}\}$ from line integral paths $l^y_{x_s}, l^y_{x_e} \in P_y$
18:            Call Algorithm 5 with $B$ and $O$ as input
19:         **end for**
20:      **end for**
21:   **end for**
22: **end while**
23: **for all** $u, v, w$ where $-d/2 \leq u, v, q \leq d/2 - 1$ **do** {Discard voxels outside the Visual Hull}
24:    **if** $O(u, v, w) = nimages$ **then**
25:       $O(u, v, w) \leftarrow 1$
26:    **else**
27:       $O(u, v, w) \leftarrow 0$
28:    **end if**
29: **end for**
```

Algorithm 5 Back-Projection Algorithm

```
1: Input a back-projection slice $B = \{row, l^y_{x_s}, l^y_{x_e}\}$ and an octree $O$
2: $z \leftarrow row$
3: Clip the space contained between $l^y_{x_s}$ and $l^y_{x_e}$ with a square of dimension $d$ to produce a polygon $F$
4: Construct a voxel span table VST $= \{y_{max}, y_{min}, Y_1, Y_2, \ldots, Y_n\}$ from $F$
5: for all $Y_j \in$ VST do
6:     $y \leftarrow j + y_{min} - 1$
7:     for all pixel span pairs $x^j_i, x^j_{i+1} \in Y_j$ do
8:         $x_s \leftarrow x^j_i$
9:         $x_e \leftarrow x^j_{i+1}$
10:        for all $x$ where $x_s \leq x \leq x_e$ do
11:            $O(x, y, z) \leftarrow O(x, y, z) + 1$
12:        end for
13:    end for
14: end for
```

4.3 Summary

We presented our Reconstruction Algorithm in this chapter. We discussed some related techniques in the literature, examining their advantages and disadvantages to assist in the development of our algorithm. We provided an overview of the octree data structure that we used to represent our reconstructed object. A description of the input data and the constraints upon this data followed. Finally we described our reconstruction algorithm.

Chapter 5

Evaluation

This chapter describes the tests that we used to evaluate our algorithms and the results of these tests. We first describe the tests and results of our segmentation algorithm. Next we describe the X-ray simulator that we developed to generate artificial X-ray data for testing our reconstruction algorithm. Finally we describe the tests and results pertaining to the reconstruction algorithm.

5.1 Evaluation of Segmentation Algorithm

In this section we describe the tests that we performed on our segmentation algorithm and the results derived from these tests.

We start by describing the data that we use as the input to our segmentation algorithm and the process of estimating the parameters that we use for our segmentation algorithm.

Next, we describe the three tests we use to evaluate our segmentation algorithm. The first is the pixel comparison test where we evaluate how well the pixels of our segmentation match those of the actual bullet. The second involves comparing the contour of the segmented bullet with that of the actual bullet. The third tests our algorithm's performance in the presence of noise.

Finally, we discuss the results of these tests.

5.1.1 Input Data

We used a set of 12 medical X-ray images to test our data. These images were not generated using a digital X-ray scanner. Instead they were created using the older, more traditional X-ray scanner that creates X-ray images on photographic material. The X-rays were scanned and converted to the BMP file format. They ranged in dimension from 1024 by 1225 to 1024 by 2004 pixels.

Our algorithm is designed to provide an automatic segmentation. In order to test our algorithm, we need to compare the automatic segmentation algorithm with a manual method.

To this end, we performed a manual segmentation of the bullets in each X-ray image. These manual segmentations provided the basis of comparison in testing our segmentation algorithm.

5.1.2 Estimation of Segmentation Model Parameters

Our Segmentation Model is based on modelling a line profile plateau using five parameters

1. $\boldsymbol{g_{min}}$: The minimum gradient for the sides of the plateau
2. $\boldsymbol{h_{min}}$: The minimum height for the sides of the plateau
3. $\boldsymbol{r_{max}}$: The maximum value of the residual average derived from a linear regression of the values on top of the plateau
4. $\boldsymbol{\theta_p}$: The angle between the linear regression and the x-axis
5. $\boldsymbol{w_{min}}$: The width of the plateau

We empirically derived the values for these parameters from the X-ray in our input set labelled XRAY03. These values were used for the other X-rays. We expect that the algorithm parameters would only need to be configured once for a particular X-ray machine since the effects that a bullet creates on an X-ray image are unlikely to vary.

g_{min} and h_{min} are the most important parameters since they are used by the segmentation algorithm to determine candidate plateaus. These candidate plateaus are then accepted or rejected based on whether they fit the remaining parameters.

g_{min} and h_{min} work together to determine candidate plateaus. A plateau is both high and steep. Many sections of the line profile fit both g_{min} and many fit h_{min} but not many will fit both. For example, a line profile section may be very steep, matching g_{min}, but also very low in height.

Correspondingly a section may be very high, matching h_{min}, but this gain in height may take place over a large section of the profile, producing a weak gradient.

By requiring the plateau to be both high and steep we discard large sections of a line profile. Since these parameters are quite stringent in combination, we set them to be fairly low in order to segment plateaus that are indistinct from the rest of the line profile. We look to the *spatial coherency* requirement of our algorithm to further discard superfluous plateaus.

We set the gradient g_{min} to a value of 2. This equates to accepting plateaus whose sides are twice as high as their width i.e. the sides of the plateau are greater than $60°$. We set the minimum height of the plateau sides h_{min} to a value of 5 when dealing with a range of intensities from $0 \ldots 255$.

r_{max} and θ_p work together to ensure that the top of the plateau occupies a range of intensities that does not drastically vary since we expect a bullet to occupy a uniform range of intensities.

We set r_{max} to 3. This parameter ensures that the values on the top of the plateau do not vary too much from the linear regression that is fitted to it. This value also indirectly ensures that plateaus that are too large are discarded: the values in these plateaus will vary too much.

We set θ_p to $45°$. This ensures that the gradient of the linear regression that is fitted to the top of the plateau is not too steep since a steep gradient implies a large change in the plateau's intensity.

We set w_{min} to 10. This parameter ensures that plateaus that are too small are discarded. This value should be tailored to match the width of the object that is being searched for.

5.1.3 Pixel Comparison

This test provides a simple pixel-by-pixel comparison between an automatic and a manual segmentation. The parameters used to perform the automatic segmentation are as follows: $g_{min} = 2, h_{min} = 5, r_{max} = 3, \theta_p = 1, w_{min} = 10$.

We iterate through each pixel in the bullet region defined by the manual segmentation. At each iteration, we test if the pixel occurring in the manual segmentation also occurs in the automatic segmentation.

The following table shows the number of pixels in the manual segmentation, the number of pixels in the automatic segmentation that occur in the manual segmentation and the corresponding percentage

X-ray Image	Number of Pixels in Manual Segmentation	Number of Pixels in Automatic Segmentation matching Manual Segmentation	Percentage of Pixels Matching
XRAY01	2431	2211	90
XRAY02	4416	4028	91
XRAY03	2425	2346	96
XRAY04	2201	1665	75
XRAY05	2331	2331	100
XRAY07	1670	1670	100
XRAY12	1661	1600	96
XRAY14	734	423	57
XRAY15	954	920	96
XRAY16	1640	1581	96
XRAY21	1637	1616	98
XRAY22	1885	1309	69

5.1.4 Contour Comparison

For our second test, we compare the distances between the contours of the manual and automatic segmentations. The parameters used to perform the automatic segmentation are as follows: $g_{min} = 2, h_{min} = 5, r_{max} = 3, \theta_p = 1, w_{min} = 10$.

Firstly, we extract the contours of the manual and automatic segmentations, representing them as two separate lists of contiguous pixels, M and A where

$$\begin{aligned} M &= \{m_1, m_2, \ldots, m_j\} \\ A &= \{a_1, a_2, \ldots, a_k\} \end{aligned} \qquad m_h \text{ and } a_i \text{ are coordinates of the form } (x, y) \tag{30}$$

To compare the distances between the contours, we iterate through each pixel in M. For each $m_h \in M$ we searched A for some a_i that would minimise d in the following metric

$$d = |m_h - a_i| \tag{31}$$

In other words, for a point $m_h \in M$ we find the point that is closest to it $a_i \in A$, and take the distance between them as the distance between the contours. For each point $m_h \in M$ we find this distance d and determine the average contour distance from all the d's that are calculated.

In the following table, we show the average distance between the contours of the manual and automatic segmentations as well as the standard deviation of the contour distance.

X-ray Image	Mean Contour Distance (pixels)	Standard Deviation of Contour Distance (pixels)
XRAY01	1.07282	0.381029
XRAY02	1.36458	0.561245
XRAY03	0.467456	0.49894
XRAY04	3.01093	3.67087
XRAY05	0.132275	0.33879
XRAY07	0.643357	0.534222
XRAY12	0.32967	0.48164
XRAY14	3.31624	3.00043
XRAY15	0.288136	0.452895
XRAY16	0.3625	0.480722
XRAY21	0.442105	0.517398
XRAY22	4.80949	5.05236

We were unable to obtain header information on the provided X-rays and are therefore unable to provide measurements in millimetres.

5.1.5 Performance in the Presence of Noise

Sensor noise is present in every operating environment. Therefore, it is important to evaluate our algorithm's performance in the presence of noise.

In order to test performance in the presence of noise, we took the XRAY05 X-ray image and added Gaussian white noise with a specified standard deviation to it. We chose to use XRAY05 because a very good segmentation was achieved with this image. By running tests on this image we are able to separate the effects of issues internal to the algorithm and the noise that affects the performance of the algorithm. Therefore, by testing with a better performing image we are able to see more clearly the effects of noise upon our algorithm.

We applied our segmentation algorithm with different noise values. The result of this is shown in the following table for Gaussian white noise with standard deviations of 1, 2 and 5.

The parameters used to perform the automatic segmentation are as follows: $g_{min} = 2, h_{min} = 5, r_{max} = 3, \theta_p = 1, w_{min} = 10$.

Standard Deviation of Gaussian Noise	Percent Pixels Matching	Mean Contour Distance (pixels)	Standard Deviation of Contour Distance (pixels)
1	99	0.037037	0.188853
2	0	N/A	N/A
5	0	N/A	N/A

When adding Gaussian Noise with a standard deviation of 2, the automatic segmentation does not detect the bullet. This is primarily due to the sensitivity of the h_{min} height parameter. By introducing noise into the line profile, many sharp inclines and declines are introduced. These inclines and declines are then accepted as candidate plateaus by the segmentation algorithm. In this process, the inclines and declines characterising the actual plateau are mismatched with other inclines and declines. They are then discarded for not meeting with the requirements of the w_{min} width parameter.

Setting the h_{min} parameter to a stricter value of 10 and reapplying the segmentation algorithm, we obtain the following results

Standard Deviation of Gaussian Noise	Percent Pixels Matching	Mean Contour Distance	Standard Deviation of Contour Distance
2	99	0.121693	0.326931

Therefore, in the presence of noise, we have to be stricter with the h_{min} parameter. However, in the presence of increasing amounts of noise, setting the h_{min} parameter to be higher does not help since the inclines and declines of actual plateaus tend to become obscured by noise.

For example, the bullet in XRAY05 generates plateaus that are approximately 20 intensity units high or more. Adding Gaussian Noise with a standard deviation of 5 introduces new inclines and declines that interfere with the inclines and declines of the actual plateau. This effect can be seen in Figure 23 and 24.

Figure 24 shows the plateau in Figure 23 with Gaussian Noise of Standard Deviation 5 added. The right side of the plateau, which is very distinct in Figure 23 has become obscured by the noise added in Figure 24.

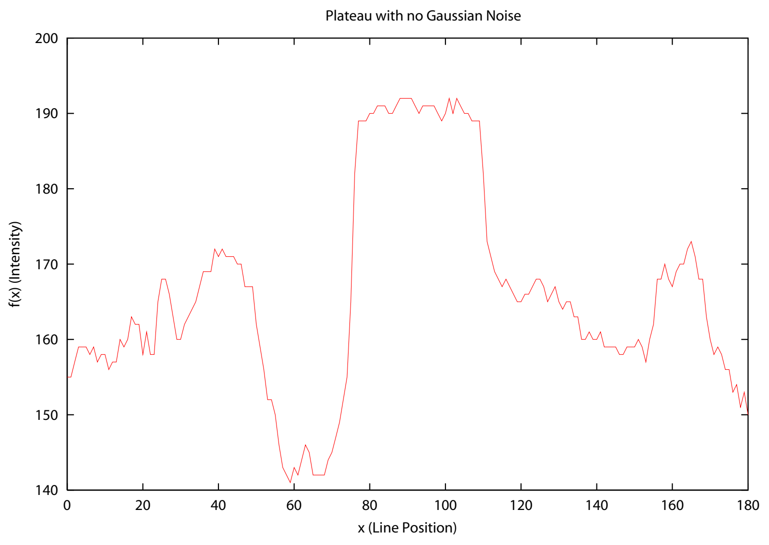

Figure 23: Bullet Plateau from image XRAY05

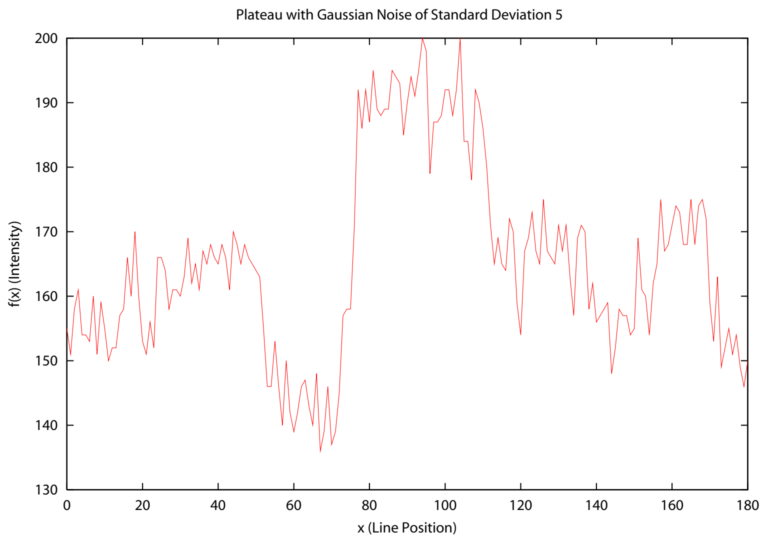

Figure 24: Bullet Plateau from image XRAY05 with Gaussian Noise of Standard Deviation 5 added

We have tested our algorithm's performance in the presence of noise quite stringently, focusing on how the parameters of the algorithm may be modified to deal with noise. The noise exhibited in Figure 24 is extreme and is not usually present in an operating environment.

Additionally, it is worth noting that the images we use for testing have already had noise introduced into them during the image scanning process to convert the X-rays to image files. Therefore, our test images have already suffered from operational noise during the actual X-ray capture as well as image scanning noise before we add artificial Gaussian Noise.

Modifying the algorithm parameters to deal with noise has limited effect and is undesirable since it reduces the accuracy of the plateau model. In practice there are better ways to deal with noise that involve performing a pre-processing step to remove noise before the main segmentation algorithm is applied. This approach is taken by Highnam et al. [18], where operational noise is removed from a mammogram to generate their h_{int} representation. To deal with extreme noise we could smooth the intensity profile with a filter, or fit a spline curve with weak continuity to the intensity profile to better estimate the general shape of the profile. We have analysed the effect of noise on our algorithm for the sake of completeness.

5.1.6 Discussion of Results

In nine of the twelve X-rays, our algorithm matched the manual segmentation by 90% or more, with an average distance contour distance of 1.3 pixels or less. This result was achieved using just one set of parameters. The X-rays were taken on a number of X-ray machines that recorded them on photographic film. They were then scanned in to create BMP image files. Considering the degradation in image quality introduced by this process, we would expect even better results on a modern digital X-ray scanner. Unfortunately we were not able to obtain X-rays of bullet wounds on such devices.

We will examine the other three cases to examine their problems and determine whether our algorithm can be improved to deal with them.

The progression of the segmentation in the area containing the bullet in XRAY04 is shown in Figure 25. We can see in panel (g) the final segmentation that the upper contour is fragmented and a leak has sprung from the lower contour. The leak is the result of thresholding within the bounding box of the dilated region shown in (f). Thus, we are accepting intensity values that fall within the intensity range , but do not lie within the region described by the intensity plateaus.

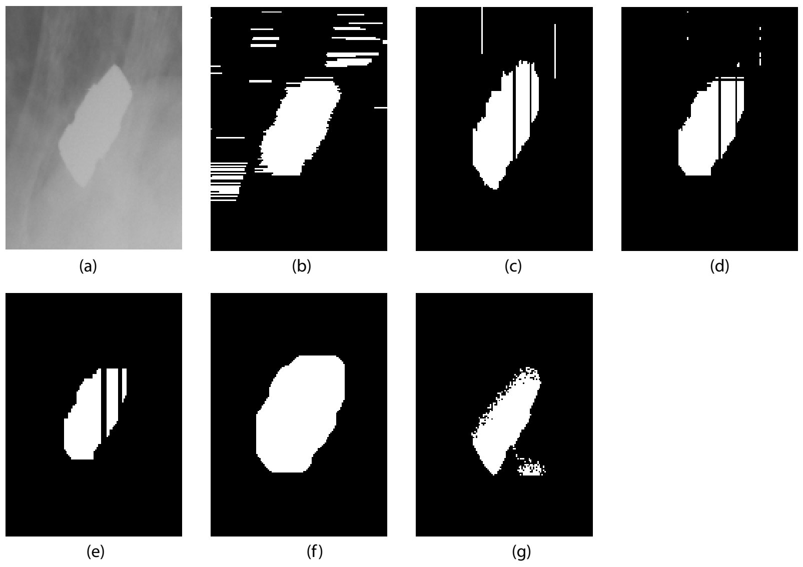

Figure 25: Progression of Segmentation Algorithm for the area containing the bullet in XRAY04 (a) Original Image (b) Horizontal Line Segments (c) Vertical Line Segments (d) Intersection of Vertical and Horizontal Line Segments (e) Eroded Image (f) Dilated Image (g) Final Segmentation

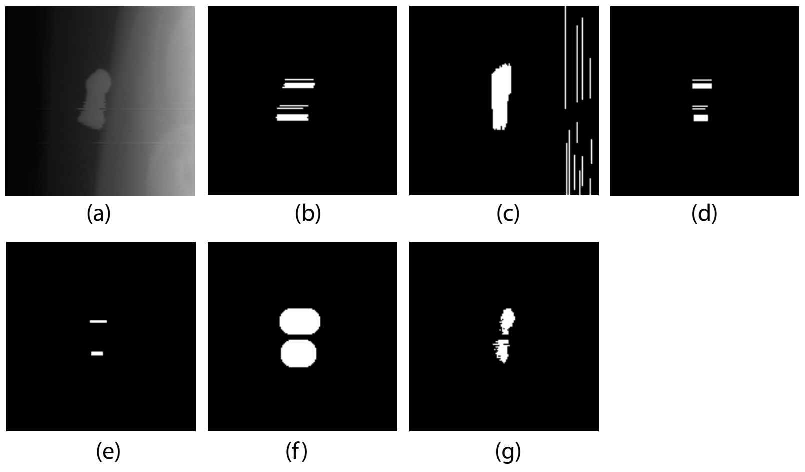

Figure 26: Progression of Segmentation Algorithm for XRAY14 (a) Original Image (b) Horizontal Line Segments (c) Vertical Line Segments (d) Intersection of Vertical and Horizontal Line Segments (e) Eroded Image (f) Dilated Image (g) Final Segmentation

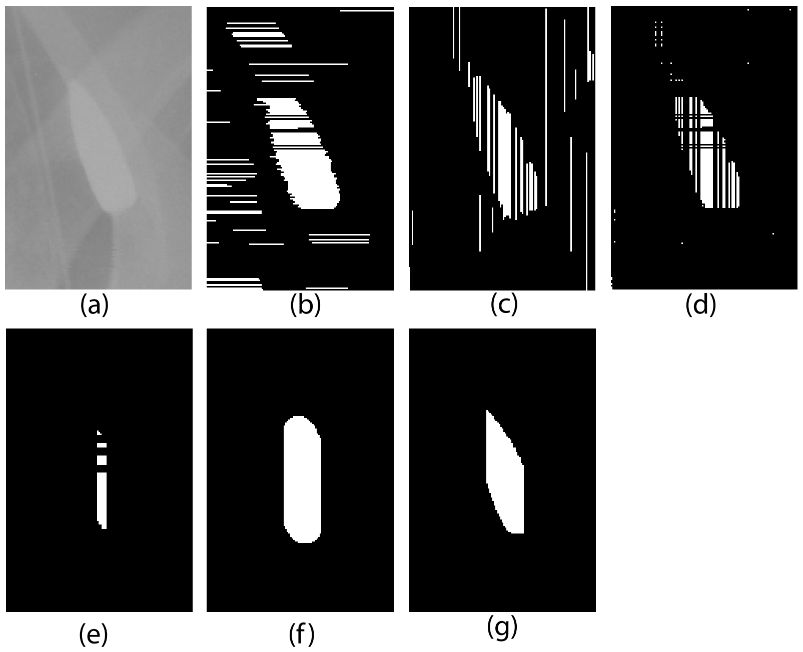

Figure 27: Progression of Segmentation Algorithm for XRAY22 (a) Original Image (b) Horizontal Line Segments (c) Vertical Line Segments (d) Intersection of Vertical and Horizontal Line Segments (e) Eroded Image (f) Dilated Image (g) Final Segmentation

The ill-defined upper contour results from the fact that the region's contour occupies a rather large range of intensities. The upper section of the contour occupies a lower range of intensities than the lower contour. Since the lower threshold is calculated from an average of the contour intensity values, the lower threshold will be greater than some of the lower contour intensity values. These contour values are then discarded during thresholding, resulting in the fragmented contour.

The progression of the segmentation in the area containing the bullet in XRAY14 is shown in Figure 26. In panel (b) of Figure 26 we can see that few horizontal plateau lines are generated. Because of this, the intersection of the vertical and horizontal plateau lines and their erosion creates two sub-regions, instead of one whole region. Therefore, the backtracking process produces two separate regions, each describing a part of the region containing the bullet. For this reason the comparison with the manual segmentation is quite poor.

By lowering the w_{min} parameter, more horizontal plateau lines would be introduced in the middle of the bullet. However, this would introduce superfluous regions into the final segmentation. A better solution would be to introduce a process after the dilation step where regions that are very close together are merged.

The progression of the segmentation in the area containing the bullet in XRAY22 is shown in Figure 27. In panel (g) we can see that the two ends of the bullet are omitted from the final segmentation. This is due to two problems: the sparsity of vertical plateaus at each end and the size of the horizontal and vertical plateaus not meeting the w_{min} parameter at the endpoints.

The sparsity of the vertical plateaus results in fragmentation of the bullet region during the intersection of the horizontal and vertical plateaus. The erosion process removes these fragmented sections, leaving the more coherent central region intact.

Unfortunately, since most of the intersected regions are eroded away, the dilated region does not fully cover the extents of the bullet, especially the endpoints. Therefore plateaus that correspond to sections of the bullet are not considered during the backtracking process, since they lie outside the bounding box of the dilatd region. This results in the endpoints being omitted from the final segmentation of XRAY22.

The orientation of the bullet also poses a problem. Since the bullet lies along the diagonal axis, the horizontal and vertical segments are very short near the endpoints of the bullet. The plateaus resulting from these segments fail to meet the w_{min} parameter and are thus discarded. Therefore any bullet lying along a diagonal will probably have sections of its endpoints trimmed off.

A naive solution might be to lower the w_{min} parameter to allow these endpoints into the solution. A better solution would take advantage of the fact that the orientation of the bullet lies along the diagonal. Far wider plateaus can be found by scanning along the northwest-southeast and northeast-southwest diagonals. Using these diagonal plateaus would result in the regions occupied by the bullet endpoints contributing to the segmentation.

5.2 X-Ray Simulator

One of the problems associated with our project was the evaluation of our reconstruction algorithm. In order to test this algorithm we required a set of X-rays of a bullet wound, taken at multiple angles. We were only able to obtain single X-rays for a number of bullet wounds. To solve this problem we implemented a simple X-ray simulator in order to provide input data for testing our reconstruction algorithm.

5.2.1 Physical Basis for Simulator

An X-ray machine projects a stream of X-rays through an object at a X-ray detector that measures how much the stream has been attenuated. For this reason, X-rays are typically modelled as line integrals that measure the degree of attenuation experienced by X-rays as they pass through an object [23]. A set of line integrals can be grouped together to form a projection.

Recall that in our background chapter we showed how the attenuation of an X-ray by an object was modelled as a line integral $P_\theta(t)$ over the object $f(x, y)$:

$$P_\theta(t) = \int_{(\theta,t)} f(x, y)\mathrm{d}s \tag{32}$$

This equation represents the sum of the values of $f(x, y)$ over a line defined by an angle θ and an offset t. This equation forms the physical basis for the implementation of our simulator.

5.2.2 Input to Simulator

Our simulator generates X-rays from two types of data. The first type is the volumetric data representing a three-dimensional range of density. We represent it as a three-dimensional array of voxels. In terms of the mathematical formulation in Equation (32) this data would define the function

$f(x, y)$. The X-rays generated by the simulator are a two-dimensional projection of the volumetric data. The value of each voxel determines the density of the volume at the coordinate of the voxel.

The second type of data is the projection data. This data defines the paths over which the densities of the volumetric data will be summed. In terms of Equation (28) each path corresponds to a line integral defined by some angle θ and an offset t. The result of this sum is stored in an accumulator which corresponds to a pixel in the X-ray image that is being created. Therefore each pixel in the simulated X-ray has a path associated with it.

We define a path $l_j^i = (\vec{a}, \vec{b})$ by a gradient $\vec{a}$ and an offset $\vec{b}$. We set $\vec{b}$ to be equal to the source of the stream of X-rays. All the paths that contribute to the creation of a single X-ray image and the pixels associated with them, $p_j = \{x_j, y_j\}$ are grouped together into a projection $P_i = \{\{l_1^i, p_1\}, \{l_2^i, p_2\} \ldots, \{l_n^i, p_n\}\}$. In order to define multiple X-rays of a "scene", multiple projections must be defined.

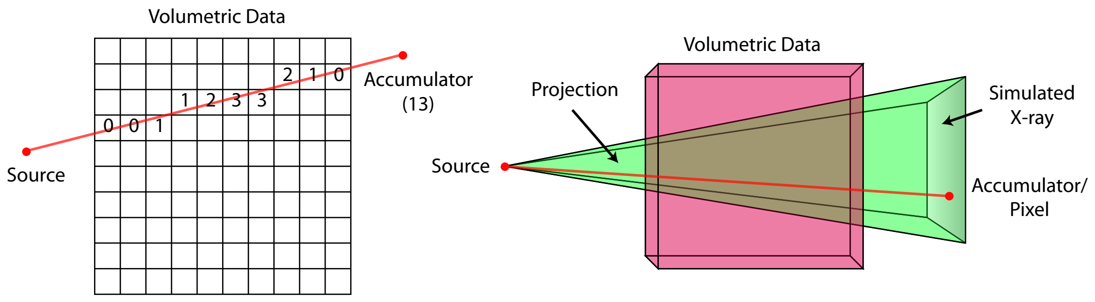

Figure 28: Generating an X-ray from volumetric and projection data. A two-dimensional and three-dimensional diagram of the process is shown here. In the 2D diagram we show how the density values of the volumetric data are summed along a path and the result is stored in an accumulator. The 3D diagram shows how this path is part of a projection that creates the X-ray and how it corresponds to a particular pixel in the X-ray.

5.2.3 Generating a Simulated X-ray

The process of generating a simulated X-ray is simple. For each path/pixel pair in a projection, the density values along each path are summed. This sum is stored in the pixel associated with the particular path. We use a three-dimensional version of the famous Bresenham line algorithm [5] to step along each voxel that lies along the path.

Once each path/pixel pair in the projection has been dealt with, the range of intensities in the simulated X-ray can be normalised and output to an image file. This process is shown in Figure 28 and described in Algorithm 6.

Algorithm 6 Algorithm for generating X-rays from volumetric and projection data

```
1: Input volumetric data into a three dimensional array $V$
2: Input projection data into $P_i$.
3: Create X-ray image $X$
4: for all $\left\{l^i_j, p_j\right\} \in P_i$ do
5:     $acc \leftarrow 0$
6:     for all $(l, m, n) \in l^i_j$ do
7:         $acc \leftarrow acc + V(l, m, n)$
8:     end for
9:     $X(x_j \in p_j, y_j \in p_j) \leftarrow acc$
10: end for
```

5.3 Evaluation of Reconstruction Algorithm

In this section we will describe the tests that we performed on our reconstruction algorithm and results derived from these tests.

Firstly, we describe the simulated data that we use as an input to our algorithm. Secondly, we describe the voxel comparison test which involves comparing the voxels of the reconstructed data with those of the simulated data. Thirdly, we describe the iso-surface comparison test which involves comparing an iso-surface of the reconstructed bullet with an iso-surface of the simulated bullet. Finally we discuss our results.

5.3.1 Input Data

To test our reconstruction algorithm we required a set of X-rays that recorded a bullet wound from multiple angles. Unfortunately, it was not possible to obtain a set of this data.

To overcome this problem, we generated an artificial set of X-rays using our X-ray simulator. The voxel data that we fed to the simulator described a 512^3 voxel volume containing an hip-bone and a

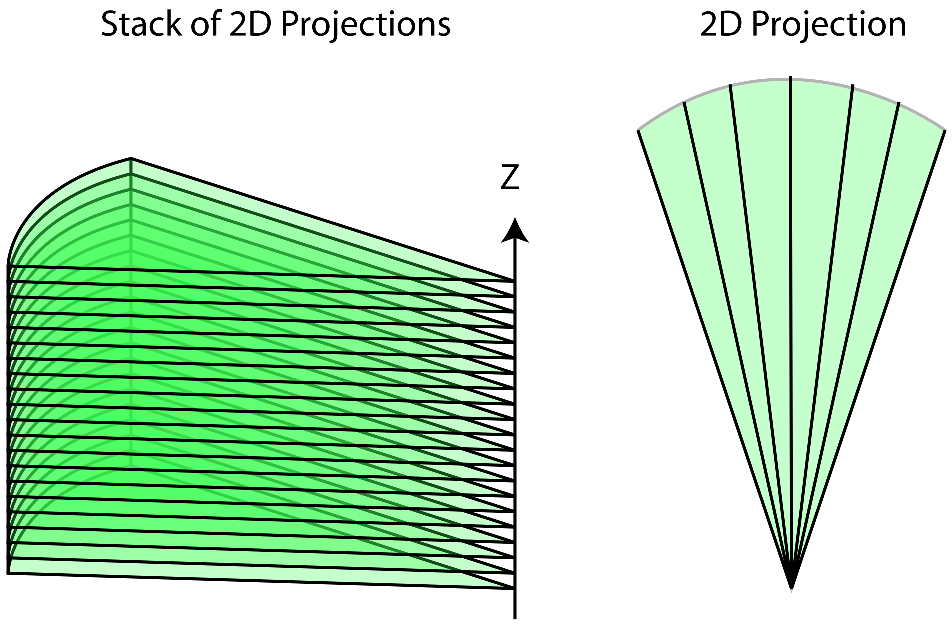

Figure 29: Configuration of Projections for Generating Artificial X-rays

bullet. This was obtained by voxelising three-dimensional hip-bone and bullet meshes, overlaying the bullet voxels on top of the hip voxels. We will refer to this voxel volume as the *simulated volume*.

From this voxel volume, we generated a set of 12 artificial X-rays. The projections used to create these X-rays were equally spaced at 30° angles around the voxel volume. The source of each projection was located 320 voxel units away from the centre of the simulated volume. Each projection consisted of a stack of 512 two-dimensional equiangular projections. These two-dimensional equiangular projections each contained 512 line integrals equally spaced within an angle of 60°. Each stack of projections therefore generated a 512 by 512 pixel image. This type of projection configuration can be seen in Figure 29.

The advantage of testing our reconstruction algorithm using artificial X-rays is that it is possible to compare the voxels of the reconstructed bullet with the voxels of the volume that was used to generate the artificial X-rays.

We applied our reconstruction algorithm using the set of 12 artificial X-rays and their associated projection data to produce a *reconstructed volume*.

5.3.2 Voxel Comparison Test

Our first test involved comparing the bullet voxels in the simulated volume with the bullet voxels in the reconstructed volume. To measure the similarity of the voxel volumes, we first counted the number of voxel locations that were occupied in either the simulated volume or the reconstructed volume. We refer to these as *comparison voxels*.

We then counted the number of voxel locations that were occupied in both the simulated and the reconstructed volume. We refer to these as *matching voxels*. We divide the number of matching voxels by the number of comparison voxels to produce a percentage. These values are shown in the following table.

Number of Comparison Voxels	Number of Matching Voxels	Percentage Match
3419	2415	70.6347%

A percentage match of 70% would seem rather low. Counting the number of voxels in the simulated and reconstructed bullet yields 2965 and 2869 voxels respectively. Thus there are 550 voxels ($3419 - 2869 = 550$) that exist in the reconstructed bullet but are not in the simulated bullet and

454 voxels ($3419 - 2965 = 454$) that exist in the simulated bullet but not in the reconstructed bullet. These 454 voxels are problematic. This is because the reconstructed bullet is a *visual hull* of the simulated bullet. According to the definition of a visual hull, it is always greater than the actual object that is being approximated. In this case, there are 454 voxels in the simulated bullet that lie outside of the visual hull that is the reconstructed bullet.

5.3.3 Iso-Surface Comparison Test

Our second test involved performing an iso-surface comparison of the simulated bullet with the reconstructed bullet. Firstly, we extracted iso-surfaces from both the simulated and reconstructed bullet voxels using the marching cubes algorithm in the VTK toolkit [41]. Secondly, we used the *Metro Mesh Comparison Tool* [12] to compare these two iso-surfaces. The Metro tool measures the Maximal, Mean and Mean Square Surface Difference as well as the Volume Difference between two meshes. These distances are numerically calculated by sampling between points on the two meshes.

We begin the definition of these differences with the ϵ function, which determines the distance between between a point p and a surface S.

$$\epsilon(p, S) = \mathbf{min}\, \mathbf{d}(p, p\prime) \qquad p\prime \in S \tag{33}$$

where **d** is the Euclidean distance between two points in E^3. The Maximal Surface Difference between two surface S_1 and S_2 can then be defined as

$$E(S_1, S_2) = \mathbf{max}\, \epsilon(p, S_2) \qquad p \in S_1 \tag{34}$$

This distance definition is not symmetric. It is not necessarily the case that $E(S_1, S_2) = (S_2, S_1)$.

Given a set of uniformly sampled distances, the Volume of the Difference between the two surfaces can be computed as the surface integral of the distance function over S_1.

$$E_V(S_1, S_2) = \int_{S_1} \epsilon(p, S_2)\mathrm{d}s \tag{35}$$

The Mean Surface Distance E_m is defined as the as the surface integral of the distance divided by the area of S_1

$$E_m(S_1, S_2) = \frac{1}{|S_1|} \int_{S_1} \epsilon(p, S_2) \mathrm{d}s \tag{36}$$

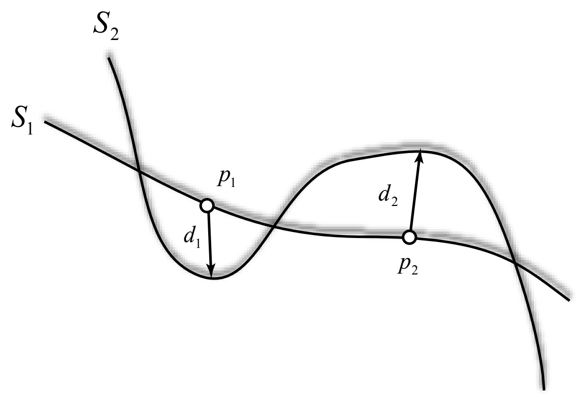

Figure 30: Illustration of signed distances between points $p_1, p_2 \in S_1$ and S_2. The distance d_1, between p_1 and S_2 is positive while d_2, between p_2 and S_2 is negative

If the surface is orientable i.e. it is possible to distinguish between the space contained within the surface and the space outside the surface, it is possible to extend the concept of surface distance between a point p of S_1 and S_2 so that the distance $\epsilon\prime$, is either negative or positive. Figure 30 illustrates this concepts with two surfaces. With this extension to the definition of surface distance, there exist two definitions for the Maximal Surface Distance, a positive and a negative definition:

$$E^+(S_1, S_2) = \mathbf{max}\ \epsilon\prime(p, S_2) \qquad p \in S_1 \tag{37}$$

$$E^-(S_1, S_2) = |\mathbf{min}\ \epsilon\prime(p, S_2)| \qquad p \in S_1 \tag{38}$$

The positive and negative equations for the Mean, Mean Squared and Volume Differences can also be defined. Metro also calculates the *total difference* between two orientable surfaces. For example, the total volume difference between S_1 and S_2 is the volume of $(S_1 - S_2) \cup (S_2 - S_1)$. Interested readers can refer to [12] for further information.

The results produced by Metro for a comparison between the simulated and reconstruction bullet iso-surfaces is shown in the following table. Metro also displays information about the bounding boxes and diameters of the meshes. The bounding box for the simulated iso-surface was $(27, 42, 17)$

by $(31, 47, 54)$ units, while the bounding box for the reconstructed iso-surface was $(27, 43, 17)$ by $(29, 48, 54)$ units.

Test Direction	Value	Bounding Box %	Diameter %
Maximal Surface Difference			
E^+	1.5456	3.9501%	4.0448%
E^-	2	5.1114%	5.234%
Mean Surface Difference			
E^+	0.6345	1.6216%	1.6605%
E^-	0.782	1.9987%	2.0467%
E^t	0.717	1.8324%	1.8764%
Mean Square Difference			
E^+	0.7952	2.0324%	2.0812%
E^-	0.8549	2.185%	2.2374%
E^t	0.8291	2.1191%	2.1699%
Volume of Difference			
V^+	8.90923e+06		
V^-	1.76483e+07		
V^t	5.17732e+07		

Here we can see more evidence that the visual hull of the reconstructed bullet does not encompass the entirety of the simulated bullet. If the reconstructed bullet encompassed the simulated bullet we would expect a value of 0 for the E^- and V^- tests since there would be no place where the reconstructed bullet iso-surface would be inside the simulated bullet iso-surface. The mean surface difference between the reconstructed and simulated mesh is 0.6345 units for E^+ and 0.782 units for E^-.

5.3.4 Discussion of Results

The results of the two tests that we have performed on our reconstruction algorithm clearly indicate an anomaly in the size of the reconstructed bullet. According to the theory that we introduced in the Background Chapter, the visual hull of a reconstructed object should always be larger than that of the original object. However, the differences between the two meshes are small: less than 1% in both the positive and negative direction.

The anomaly with regards to the size of the reconstructed object can be attributed to rounding errors within our reconstruction algorithm. We use Bresenham's line algorithm to step through the voxel volume. Bresenham's algorithm discretises points on a line into voxel locations. This discretisation introduces rounding errors into the reconstruction.

5.4 Conclusion

In this chapter we presented the evaluation of our segmentation and reconstruction algorithms. For each test, we described the input data to the test, the method that was used to test the algorithm and the results of the test. We also discussed the results of each test noting problems and suggesting solutions.

Our segmentation algorithm performed well on a set of 12 X-rays, achieving a pixel match of 90% or more with the manual segmentation on nine of these X-rays. The mean contour distance between the automatic and manual segmentation 1.3 pixels or less, with a standard deviation of 0.56 pixels or less. For the remaining three X-rays the pixel match was 57%, 69% and 75% with respective mean contour distances of 3.3, 4.8 and 3.0 pixels. These are good results, considering the quality of the data that the algorithm operated on:

1. Not all of the X-rays were taken on the same machine.
2. The X-rays were recorded on photographic film.
3. The X-rays were scanned in from the photographic film.

We would expect even better results using X-rays taken on modern digital X-ray scanners. Our segmentation algorithm is fairly sensitive to noise. This sensitivity can be ameliorated by smoothing intensity profiles, or fitting splines to the profiles.

The test case that we used to evaluate our reconstruction algorithm achieved a 70% voxel match and a 0.7952 positive mean surface difference. Our reconstruction algorithm produced anomalous behaviour by producing a reconstructed bullet that was smaller than the simulated bullet. Since the reconstructed bullet is a visual hull, it should be larger than the simulated bullet. This can most likely be attributed to rounding errors introduced by using the discrete Bresenham stepping algorithm.

Chapter 6

Conclusion and Future Work

The goal of this thesis was to develop a technique to identify the three-dimensional shape and location of a bullet from multiple X-ray projections. To accomplish this, we based our technique on two techniques: Image Segmentation and Volume Carving.

The Image Segmentation section of our project involved developing an algorithm to identify bullets within X-rays. The basis for our segmentation technique centres around the intensity plateaus generated by bullets in line profiles taken across X-ray images.

We developed a model to define and identify these plateaus within X-ray images and used this model to develop a segmentation algorithm. This algorithm searches for line profiles within the image that fit the plateau model. Regions containing a large confluence of these profiles are selected as bullet candidates. The range of intensities occupied by these profiles are used to segment the bullet out of the image.

We base our Volume Reconstruction technique on a simple voxel-based implementation of *Volume Carving* [2]. An octree voxel volume is used to represent the reconstructed three-dimensional shape and position of the bullet. The reconstruction process involves reconstructing two-dimensional slices of this three-dimensional volume.

In order to reconstruct a two-dimensional slice, we create back-projection volumes from the bullets that we segment from the input X-Rays. These back-projection volumes are intersected with the two-dimensional slice. This involves marking the number of back-projection volumes that intersect with a voxel. A voxel that intersects with all the back-projection volumes input to the algorithm is considered to be within the visual hull of the reconstructed bullet. Only voxels that lie within the visual hull are accepted as contributing to the final reconstruction.

6.1 Summary of Results

Evaluation of Segmentation Algorithm: We tested our algorithm on a set of 12 X-rays. For nine of these X-rays, the algorithm matched over 90% of the pixels contained within the bullet. In the nine X-rays, the difference between the contour of the segmentation and the actual bullet was on average less than one pixel. In the other three X-rays, the algorithm matched 59%, 69% and 79% of the pixels contained within the bullet respectively.

The three X-rays that do not perform so well reveal interesting issues in our algorithm. Firstly, we perform a thresholding operation in the region containing the bullet. This region is represented with a bounding box. Since this bounding box does not always accurately follow the shape of the bullet, pixels that are not in the bullet may be segmented out.

Secondly, the minimum width parameter (w_{min}) of our segmentation algorithm can cause problems if sections of the bullet are very thin. This parameter exists to discard superfluous information during the segmentation process.

Thirdly, we scan for bullets in the horizontal and vertical directions. This can be problematic when the bullet lies along a diagonal axis because the endpoints of the bullets will consist of short horizontal and vertical lines. If these lines are too short they will be truncated by the w_{min} parameter resulting in the endpoints of the bullets being “trimmed”.

Solutions to some of these problems have been suggested in our future work section.

From these results we conclude that we have successfully researched and implemented a bullet segmentation algorithm. Our algorithm is successful for three reasons:

- We achieved a segmentation of over 90% on nine of the twelve X-rays we used to test our algorithm
- We used one set of segmentation parameters for all twelve X-rays. These X-rays were taken on different X-ray machines with different configuations. As we have stated in our Segmentation Chapter, one can configure the segmentation algorithm parameters once for a particular X-ray machine. This is because the intensity profile of lead will vary only slightly due to the high

attenuation experienced by X-rays interacting with lead. This significantly reduces the user intervention required to operate the algorithm.

- The X-ray data that we used to test our algorithm was not high quality. It was generated by older X-ray scanners that recorded X-ray images on photographic film. These images needed to be scanned in to image files for use by our algorithm, increasing the amount of noise experienced by our algorithm. We expect that even better results could be achieved using data captured from digital X-ray scanners.

Evaluation of Reconstruction Algorithm: To test our reconstruction algorithm, we artificially generated a set of 12 X-rays using the X-ray simulator that we had developed. These X-rays showed a simulated bullet overlayed on a hip bone. The reconstructed bullet matched 70% of the voxels in the simulated bullet. The mean surface difference between the reconstructed and simulated bullet was less than 1 unit. However, rounding errors in the reconstruction algorithm resulted in some voxels belonging to the simulated bullet not being included in the reconstructed bullet.

This behaviour is anomalous since the reconstructed bullet is a visual hull of the simulated bullet and thus should include all parts of the simulated bullet. It can be attributed to rounding errors associated with our use of the Bresenham line stepping algorithm which discretises points on a line. For these reasons the implementation of our reconstruction algorithm was moderately successful.

In conclusion, we have achieved our aims of developing a segmentation algorithm and implementing a reconstruction algorithm that reconstructs the shape and position of bullets from multiple X-rays.

6.2 Future Work

There are a number of areas in which our work could be extended.

6.2.1 Creating a Bounding Polygon for use during the thresholding phase of the Segmentation Algorithm

One of the problems associated with our segmentation algorithm is that pixels that fall outside the bullet may be included in the thresholding operation that extracts the bullet pixels. This is because a bounding box is used to bound the range of pixels that are being thresheld out of the image. The bounding box will contain pixels that do not lie within the actual bullet most of the time. However, some of these pixels may fall within the range of intensities that are being thresheld and will thus be included within the final segmentation.

To deal with this problem, a more precise bounding polygon could be constructed from the endpoints of the line profiles that contribute to a region containing a bullet.

6.2.2 Thresholding different ranges of intensity for different parts of the bullet

An average lower value is calculated for the purposes of thresholding the bullet out of the general region that it occupies. This value usually corresponds to the pixel intensities on the contour of the bullet. However, this value may vary over the bullet's contour. This results in sections of the contour being badly fragmented or left out. By allowing the lower threshold value to vary depending on the area of the bullet that is being segmented, this loss of information could be avoided.

6.2.3 Diagonal Plateau Scanning

At present our segmentation algorithm only scans for plateaus in the horizontal and vertical direction. When a bullet lies along a vertical axis, the endpoints of the bullet may be truncated by our algorithm since the width of the horizontal and vertical plateaus in these regions is too short. By adding plateau scanning along diagonal axes, more significant and wider plateaus can be detected.

Bibliography

[1] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, Jun 1994.

[2] Laurentini Aldo. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, February 1994.

[3] N. Alexandris and A. Klinger. Picture decomposition, tree data structures and identifying directional symmetries as node combinations. *Computer Graphics and Image Processing*, 8:43–77, 1976.

[4] B.G. Baumgart. *Geometric Modelling for Computer Vision*. PhD thesis, Stanford University, 1974.

[5] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[6] P. Brodatz. *Textures - A Photographic Album for Artists and Designers*. Dover, New York, 1966.

[7] M.S. Brown, L.S. Wilson, B.D. Doust, R.W. Gill, and C. Sun. Knowledge-based method for segmentation and analysis of lung boundaries in chest x-ray images. *Computerized Medical Imaging and Graphics*, 22(6):463–477, Nov-Dev 1998.

[8] C. Buehler, W. Matusik, L. McMillan, and S. Gortler. Creating and rendering image-based visual hulls. MIT LCS Technical Report 780, Massachusetts Instute of Technology, March 1999.

[9] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[10] J.C. Carr. 3d shape reconstruction using volume intersection techniques. CUED/F-INFENG/TR 300, University of Cambridge, September 1997.

[11] C.H. Chien and J.K. Aggarwal. Volume / surface octrees for the representation of three-dimensional objects. *Computer Vision, Graphics and Image Processing*, 36(1):100–113, October 1986.

[12] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. Technical report, Istituto per l'Elaborazione dell'Informazione - Consiglio Nazionale delle Ricerche, Pisa, Italy, 1998.

[13] L.D. Cohen and I. Cohen. Finite-element methods for active contour models and balloons for 2-d and 3-d images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, November 1993.

[14] T.F. Cootes and C.J. Taylor. Active shape models - 'smart snakes'. In *Proceedings of the British Machine Vision Conference*. Springer-Verlag, 1992.

[15] T. Feldman. Generating iso-value contours from a pixmap. In D. Kirk, editor, *Graphics Gems III*. Academic Press, 1992.

[16] R. Gupta and P.E. Undrill. The use of texture analysis to delineate suspicious masses in mammography. *Physics in Medicine and Biology*, 40:835–855, 1995.

[17] R. P. Highnam, M. Brady, and Shepstone B. J. A representation for mammographic image processing. *Medical Image Analysis*, 1:1–19, 1996.

[18] R. P. Highnam, M. Brady, and Shepstone B. J. The h_{int} representation and calcifications. In *Proceedings of the First UK Medical Image Processing Conference, Oxford*, July 1997.

[19] G.N. Hounsfield. A method of and apparatus for examination of a body by radiation such as x-ray or gamma radiation. Patent Specification 1283915, The Patent Office, 1972.

[20] J. Ivins and J. Porrill. Everything you always wanted to know about snakes (but where afraid to ask). Technical report, Artificial Intelligence Vision Research Unit, University of Sheffield, 1993.

[21] J. Ivins and J. Porrill. Statistical snakes: Active region models. In *British Machine Vision Conference*, pages 377–386, Sep 1994.

[22] J. Ivins and J. Porrill. Active-region models for segmenting textures and colors. *Image and Vision Computing*, 13(5):431–438, May 1995.

[23] Avinash C. Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[24] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331, 1987.

[25] A. Klinger and C.R. Dyer. Experiments in picture representation using regular decomposition. *Computer Graphics and Image Processing*, 5:68–105, 1976.

[26] T.S. Lee, D. Mumford, and A. Yuille. Texture segmentation by minimizing vector-valued energy functionals: the coupled-membrane model. *Proceedings of the European Conference on Computer Vision*, 588:165–173, 1992.

[27] L.R. Long and G.R. Thoma. Segmentation and feature extraction of cervical spine x-rays images. In *Proceedings of SPIE Medical Imaging 1999: Image Processing*, volume 3661, pages 1037–1046, Feb 1999.

[28] J. Lotjonen, I.E. Magnin, J. Nenonen, and T Katila. Reconstruction of 3d geometry using 2d profiles and a geometric prior model. *IEEE Transactions on Medical Imaging*, 18:992–1002, 1999.

[29] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society*, pages 187–217, 1980.

[30] W.N. Martin and J.K. Aggarwal. Volumetric description of objects from multiple views. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 5(2):150–158, 1983.

[31] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of 12th Eurographics Workshop on Rendering*, pages 116–126, 2001.

[32] R.D. Merrill. Representation of contours and regions for efficient computer search. *Communications of the ACM*, 16(2), feb 1973.

[33] A. Mohammed-Djafari and C. Soussen. Multiresolution approach to the estimation of the shape of a 3d compact object from its radiographic data. In *Proceedings of the SPIE, Mathematical Modelling, Bayesion Estimation and Inverse Problems*, pages 150–160, July 1999.

[34] T. Nopola, A. Järvi, E. Svedström, and Nevalainen O. Segmenting bones from wristhand radiographs. TUCS Technical Report 371, Turku Centre for Computer Science, December 2000.

[35] N. Paragios and R. Deriche. Geodesic active regions for texture segmentation. Rapport da Rerche 3440, INRIA, June 1998.

[36] J.R. Parker. *Algorithms for Image Processing and Computer Vision*. John Wiley & Sons, Inc., 1997.

[37] Sylvain Petitjean. A computational geometric approach to visual hulls. *International Journal of Computational Geometry and Applications*, 8(4):407–436, 1998.

[38] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence. *Computer Vision, Graphics and Image Processing*, 40:1–29, 1987.

[39] A. Rosenfeld and A.C. Kak. *Digital Image Processing*. New York: Academic, 1976.

[40] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2), Jun 1984.

[41] W. Schroeder, K. Martin, and B. Lorenson. *The Visualization Toolkit: An Object-Orientated Approach to 3D Graphics*. Kitware Inc., third edition, 2003.

[42] A. Siebert. Dynamic region growing. *Vision Interface*, 1997.

[43] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. PWS Publishing, second edition, 1999.

[44] C.L. Stong. *The Scientific American Book of Projects for The Amateur Scientist*. Simon and Schuster, 1960.

[45] R. Szeliski. Rapid octree construction from image sequences. *Computer Vision, Graphics and Image Processing: Image Understanding*, 58(1):23–32, jul 1993.

[46] George Thimm. Segmentation of X-ray image sequences showing the vocal tract (with tool documentation). IDIAP-RR 1, IDIAP, January 1999.

[47] J. Veenstra and N. Ahuja. Efficient octree generation from silhouettes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 537–542, jun 1986.

[48] Ge Wang, Ping-chin Cheng, and M. W. Vannier. Spiral ct: Status and future directions. Technical report, Deparment of Radiology, University of Iowa, 1900.

[49] M. Winter. Web elements periodic table. Available at **http://www.webelements.com**, 2004.

[50] M. Yam, R. P. Highnam, and M. Brady. Detecting calcifications using the h_{int} representation. In *Proceedings of the 13th International Conference on Computer-Assisted Radiology and Surgery, Paris*, Jun 1999.

[51] D. Ziou and S. Tabbone. Edge detection techniques - an overview. Technical Report 195, Universit de Sherbrooke, 1997.