Fluoroscopy Based Guidance for Percutaneous Cement Injection in Minimally Invasive Hip Refixation

Bas van den Berg

Abstract—At the Leiden University Medical Centre (LUMC) a novel method for refixating hip prostheses is being developed. A hip prosthesis can loosen due to bone resorption. This creates fibrotic tissue adjacent to the cement, which is holding the prosthesis in place. Instead of replacing the complete hip prosthesis, new cement is injected in the area which contains the fibrotic tissue. This refixates the prosthesis and extends its lifetime. Currently a CT scanner is used during the procedure to position needles which are used to inject the cement. Monitoring cement injection can be a difficult task, because dense materials like the hip prosthesis obstruct the view on the fluoroscope. To improve the reliability and reduce the procedure time we have developed a software tool to help the surgeon plan and verify the cement injection. This tool can simulate the view on the fluoroscopy C-arm, based on preoperative CT data. The Digitally Reconstructed Radiographs (DRRs) can be augmented with cement, needles and tissue overlays. The main goal is twofold. First of all we aim to provide the surgeon with an optimal planning for the viewing angle of the fluoroscopy C-arm. Secondly we want to provide the ability to perform postoperative analysis and verification of the cement injection. We can provide optimal angles based on the visibility of needles, accuracy of the detectable needle position and the certainty of the cement filling of the target area. We also provide designs for needle attachments to make the tilting angle of a needle more visible on the projection images recorded by a fluoroscope.

1 Introduction

When people suffer from arthritis or severe damage to their hip joint they can be helped by replacing the hip joint with an artificial variant. The orthopaedic procedure where the hip joint is replaced with a prosthetic implant is called Total Hip Arthroplasty (THA). The hip joint replacement restores functionality and relieves pain. Even though this procedure has a high success rate these prostheses do not last forever. Annually there are 350,000 hip prostheses that have to be revised [Malchau et al., 2000]. The primary cause of prosthesis loosening is fibrotic tissue forming in the space between a prosthesis and the bone in which it is fixated. The fibrotic tissue is too soft to keep the prosthesis in place and thus the prosthesis becomes loose. This causes pain to the patient and limits his or her mobility. The conventional solution is to replace the prosthesis, but this surgical procedure has a higher risk than the primary hip replacement. Thus for patients that are physically in a weak condition, such as many elderly people, this risk is often too large. Each year, 100,000 patients are unable to undergo the hip replacement procedure.

A novel method for refixating loosened hip prostheses has been developed at Leiden University Medical Centre (LUMC) by Poorter et al. [2008]. New cement is injected at the position where the fibrotic tissue has formed, see figure 1 for an overview. This method is currently performed with both CT and fluoroscopy guidance for needle placement and cement injection. This procedure currently lacks a proper planning system and it is difficult to monitor the progress of the procedure. As of December 2012 there are only three places in the world where this procedure is performed, at the LUMC in the Netherlands, KU Leuven in Belgium and Halifax Health in Canada.

Existing Computer Aided Surgery (CAS) systems often use 3D volumes obtained from CT or MRI to perform preoperative planning, such as used by Bifulco et al. [2002]; Guéziec et al. [1998]; Weese et al. [1997] and mentioned by van der Bom et al. [2011]; Markelj et al. [2008]; Zöllei et al. [2001]. In orthopaedic surgery, using stereoscopic optical marker based tracking as used by Merloz et al. [2007]; Nolte et al. [2000] is a common way to ensure the preoperative plan is being followed intraoperatively. Optical tracking systems work with reflective markers that are fixed to the patients bones. Markers are also applied to the surgical instruments and all markers are tracked with a combination of infrared light and infrared cameras. Brainlab and Zimmer are well known providers of these kind of systems. Although these systems are accurate they are also expensive and space and time consuming in set up and use. Our target group is one which cannot undergo a major surgery, thus attaching the markers to the patient’s bones is not desired.

Another commonly researched option is to use fluoroscopy imaging [Guéziec et al., 1998; de la Fuente et al., 2005; Bifulco et al., 2002; Lemieux et al., 1994; Merloz et al., 2007; Nolte et al., 2000; Suhm et al., 2000] to compare the preoperative plan with the intraoperative situation. The intraoperative fluoroscopy image is registered to either preoperative fluoroscopy images or a preoperative CT volume. The advantage over optical tracking is that you do not need to attach markers, causing it to be less invasive. However the tracking algorithms are more complex and the radiation exposure makes it less

1. www.brainlab.com
2. www.zimmer.com
We have developed a planning system and verification tools to ensure that the minimally invasive hip refixation procedure is more consistent and verifiable. Considering the different tracking options and the current setup, we decided to focus on using just a fluoroscope for guidance during the procedure. No longer requiring a CT scanner intraoperatively, the procedure not only has the potential to become faster and cheaper, it will also be easier to use in existing operating rooms. Our planning and verification software provides: A simulation of the fluoroscopy view from a preoperative CT volume, both with and without needles and injected cement; Verification of the injected cement by reconstructing the cement distribution from one or multiple fluoroscopy images. Because we use only a fluoroscope during the intervention the information about the out of plane position and rotation of needles has been partly lost. To address this problem, we propose and evaluate adopted needle shapes by means of attachments. The adopted needle shapes make it possible to increase the visibility of out of plane rotations (tilting) of a needle. The user interface and visualization for both the planning and the cement reconstruction have been developed simultaneously by Renata Raidou. To summarize our contributions:

- Planning and verification software.
  - A simulation of the fluoroscopy view from a preoperative CT volume. Including the option to simulate:
    * Inserted needles.
    * Injected cement.
  - Verification of the injected cement. Done by reconstructing the cement distribution from:
    * One fluoroscopy image.
    * Multiple fluoroscopy images.
- Needle attachments, increasing the visibility of out of plane rotations of needles.

2 Background on Minimally Invasive Hip Prosthesis Refixation.

We will give a more detailed description of the procedure developed by Poorter et al. [2008] to refixate loosened hip prostheses mentioned in section 1. Usually people that suffer from reduced mobility and pain due to loosening of the prosthesis are considered for a prosthesis revision. This is a risky procedure and elderly people do not always have the health to endure this. The new refixation method provides an opportunity for those people not fit enough for prosthesis revision.

Instead of replacing the whole hip prosthesis the prosthesis is refixated by injecting extra bone cement into the cavity formed by bone reabsorption. The injection is done by inserting hollow needles that reach into the bone. Preoperatively the surgeon determines where the cement has to be injected using a CT volume. With the help of this information is also decided where the needles will be inserted. There is currently no way to take this information into the operating room. To make this method more accessible we propose a new work flow where the CT scanner is no longer needed in the operating room and where we provide a step by step plan with images of what should be seen on the fluoroscope during the procedure.

3 Introduction to Fluoroscopy Guidance

A fluoroscopy C-arm is a common option for intraoperative guidance. It is a relatively small device compared to CT or MRI tracking devices, is not intrusive unlike optical tracking, does not need special preparation and can easily be manipulated into the desired position. It also has the advantage that it can be used for continuous recording. The disadvantages are radiation exposure (compared to optical tracking) and two instead of three dimensional images (compared to a CT scanner). It is however possible to combine multiple fluoroscopy images into a three dimensional data volume. Aligning the recorded images can range from trivial to complicated depending on how much is known of the recorded situation. When the pose of the fluoroscope is known during the different recordings the reconstruction can be done directly with conventional methods. When the pose of the projection images is not (entirely) known the pose has to be found first. One general way to do this is by using a preacquired volume dataset and reconstruct the pose (position and orientation) of the dataset according to the acquired projection image(s). In general the pose is found in an iterative way by matching the acquired fluoroscopy projection image with a Digitally Reconstructed Radiograph (DRR) generated from a CT volume.
4 Related Work in Fluoroscopy Guidance

There are various methods to align fluoroscopy images with the intraoperative situation. Comparisons between some methods can be found for example in the papers by Markelj et al. [2010]; Roche et al. [1999]; Zitová and Flusser [2003]. There is a relatively new method that stands out. It uses a special fluoroscopy C-arm that can also acquire Computed Tomography Fluoroscopy (CTF) volumes. The C-arm then acts like a CT scanner and rotates around the patient acquiring images and reconstructing a volume. The C-arm basis stays in place so that with every recording the relative pose to the other recordings is known and there is no need for a matching algorithm. This also applies to the later acquired projection images in this method. The pose relative to the reconstructed volume is known and thus the images can directly be used to augment the reconstructed volume.

One commercial product that uses this method is the XperGuide by Phillips, which requires a special fluoroscope also by Phillips. This method is designed for needle insertions, mostly for the purpose of biopsies. The XperGuide is described by Racadio et al. [2007]; Ruijters et al. [2008]; Leschka et al. [2011]; Spelle et al. [2009]. In short, it first acquires a CTF dataset and then picks viewing angles for both needle placement and insertion. For the needle placement a view is chosen which makes the insertion direction of the needle well visible. For the needle insertion the view shows the progress of the insertion. The XperGuide can then automatically position the C-arm at the chosen viewing angles. For the placement it has a 'top down' view and during insertion it shows a side view of the needle to monitor the insertion progression. The situation for both views is shown in figure 2. This system is designed for biopsies and fails when applied to our procedure. Any obstructions by bone or worse a prosthesis are not taken into account. In case we were to use this system for our procedure we will be unable to see anything during the placement of the needle because the prosthesis will block the needle insertion spot as depicted in figure 2. So when placing a needle for insertion it will be impossible to see if we are at the correct position. During the actual insertion of a needle the side view will likely not cause any problems, because the prosthesis won’t be blocking the view. Apart from the problems in verifying the locations to insert needles we are also interested in monitoring the cement injection. There is currently no method to determine the best way to monitor the cement injection.

5 Methods

We propose to improve the workflow of the method developed at the LUMC so that it requires only fluoroscopy, not intraoperative CT or optical tracking. We aim to provide the surgeon with better insight in the intraoperative situation. To accomplish this we developed a planning and verification software tool. The resulting planning workflow is shown in figure 4. This software provides a 3D view of the CT volume to place needles and inspect the position/situation of presegmented tissues (shown in figure 3) and simulation of fluoroscopy images with several features. These features include the visualization of needles, cement and interactive overlays of the presegmented tissues. Additionally it is also possible to visualize the predicted visibility of the injected cement, which can also be used for verification of the injected cement postoperatively. This paper deals with the algorithms behind the functionality, but also contributes to needle visibility by increasing the sensitivity to out of plane rotations (tilting). Thus we provide:

- Digitally Reconstructed Radiograph generation (section 5.1), including:
  - Needle and cement simulation (section 5.1.2).
  - Overlays of segmented tissue (section 5.1.3).
- A method to evaluate of the amount of injected cement (section 5.2).
- Volume reconstruction of the injected cement, including the percentages of filling (section 5.3), by means of:
Figure 3. These figures show the main window in the planning view of HipRFX. In figure (a) only the slice viewer is enabled, showing the raw CT data with adjustable window and level. In figure (b) the same dataset is shown, but this time a segmentation file is loaded which enables us to show several structures such as: old cement, new cement, the cortical bone and the prosthesis. We also have automatically segmented the skin and we can show an interactive isovalue render of the CT data to quickly visualise the bony structures or other high contrast elements such as the prosthesis.

- Backprojection (section 5.3.1).
- Algebraic Reconstruction (section 5.3.2).

- Fluoroscopy viewing angle optimization (section 5.4) with:
  - Different viewing angle scores (section 5.4.1).
- Enhancements to the needle to increase the sensitivity to tilting (section 5.5).

The goal is to provide the surgeon with the ability to make a plan for the procedure, determine the optimal viewing angles, visualize all the steps and finally verify the results afterwards. Our proposed workflow includes a step by step plan with fluoroscopy images that can be used to verify against the intraoperative situation. We also provide an optimization method which can determine the optimal viewing angle for the fluoroscopy C-arm for each step of the procedure, such that the surgeon can have the best view on the situation at all times. A diagram depicting the workflow is shown in figure 4.

To visualize what the surgeon should see on his fluoroscope during the procedure we have to be able to simulate fluoroscopy images. This can be done by making Digitally Reconstructed Radiographs (DRRs). With these we can simulate the view on the fluoroscope for the different situations and steps in the procedure. This includes not only simulating a fluoroscopy view from the preoperative CT scan, but also simulating needles and cement that can be inserted into the volume. The simulation is done by approximating the physical X-rays through ray casting on the CPU.

Figure 4. The planning workflow for minimally invasive hip prosthesis refixation procedure in a block diagram. First a CT scan is acquired on which the planning can be based. During the planning needle insertion locations can be determined as well as the viewing angles for the C-arm fluoroscope. For all steps the simulated fluoroscopy images can be saved so that they can be compared to the real situation intraoperatively. Afterwards the amount of cement injected can be verified from intraoperative fluoroscopy images.

5.1 Generating DRRs

For the purpose of simulating the procedure and providing a way to relate the preoperative plan to the intraoperative situation we use Digitally Reconstructed Radiographs (DRRs). With these we can simulate the view on the fluoroscope for the different situations and steps in the procedure. This includes not only simulating a fluoroscopy view from the preoperative CT scan, but also simulating needles and cement that can be inserted into the volume. The simulation is done by approximating the physical X-rays through ray casting on the CPU.
For generating DRRs we only need the intensity absorption term. This leaves:

$$I(D) = I_0 e^{-\int_0^D \tau(t)dt}$$

which is identical to the X-ray attenuation integral equation (1).

For practical application it is common practice to evaluate the volume rendering integral numerically, as done by Botha and Post [2002]; Alakuijala et al. [1996]; Sherouse et al. [1990]. Commonly a discretization of a ray-casting integral would result in an equation similar to:

$$I(D) = I_0 \prod_{i=1}^{D/\Delta s} e^{-\tau(s_i)\Delta s}.$$  

(4)

Here $\Delta s$ being the distance step. It is possible to substitute $\tau(s_i)\Delta s$ with an opacity value, $\alpha(s_i)$, corresponding to the $i$th segment along the ray. Because we do not have to take into account any colouring or shading we can directly discretize the exponent in equation (3), resulting in:

$$I(0) = I_0 D e^{-\sum_{i=0}^{D/\Delta s-1} \alpha(s_i)}$$

(5)

Note that for the calculation we have not made any approximations except for the discretization.

5.1.2 Simulating Needles and Cement

So far we have only made DRRs simulating the moment the CT scan was made. It would be desirable if we could more accurately represent what the fluoroscopy view will look like during the refixating procedure. Being able to show inserted needles in DRRs would help accomplish this.

To augment a DRR image with more information we can augment the source of the DRR, which is the CT volume. By substituting the current attenuation values with the attenuation values of the new object and generating a DRR from this augmented volume we can effectively simulate the new situation. To do this, we discard the information that was previously in the replaced voxels. During an actual cement injection procedure the material would not be replaced but displaced, which is not taken into account here. Because we are replacing voxels representing a material with a low attenuation coefficient with material with a high attenuation coefficient, the loss in attenuation due to discarding the original voxel values is minimal. However it should be kept in mind in case we want to replace parts of the volume with different attenuation properties as it could potentially lead to an incorrect simulation of the situation.

5.1.3 Overlays in the DRRs

To be able to provide additional information about the location of the different structures we have integrated the ability to show mask data in the dynamically generated DRR. An example is shown in figure 6. The overlays correspond to the segmentation shown in figure 3b. At
the same time as casting a ray through the CT volume we cast a ray through the mask data volume. The mask consists of integer data with every value representing a different tissue type. For each mask, $m$, we accumulate the corresponding opacity, $\alpha$, to get the total opacity

$$T(m, 0) = T_D e^{-\sum_{i=0}^{D/A-1} \alpha_m(s_i)} \quad (6)$$

Then we blend all the mask colours depending on their total opacity

$$C_t(0) = \frac{\sum T(m, 0) \cdot C(m)}{\sum T(m, 0)} \quad (7)$$

Both the colour and the opacity can be set by the user per mask. Setting the opacity to a low value can give extra information about the thickness of the particular structure or tissue that is represented by this mask.

### 5.2 Calculating the Cement Filling Percentage

The Cement Filling Percentage indicates what percentage of the predefined cement target volume is filled with cement. We can obtain this information using three fluoroscopy images: One without cement, one with the cement target volume filled and one displaying the final filling. The first two images can be simulated and the third image is recorded intraoperatively. An example set of these images is shown in figure 7. Because the cement contains a contrast agent it is well visible in the fluoroscopic images. Places that get filled with cement have a higher attenuation value and will appear darker on the projection image than without the cement. To quantify the amount of cement added to the volume we can compare two fluoroscopy images, one with cement and another from the same angle without cement. For example figure 7b and figure 7a or figure 7c and figure 7a. We can then assign a percentage value to every pixel indicating how much of the cement target volume covered by that particular pixel is filled by the cement.

So how do we compare them to retrieve the amount of cement? In section 5.1 we had the equation for the X-ray attenuation equation (1), repeated here for convenience

$$I(D) = I_0 e^{-\int_0^D \mu(s, E) ds}.$$

Assuming that the amount of cement added has a linear relation to the increase in attenuation coefficient we can deduce the percentage of the target volume that is filled with cement. This percentage would be calculated as

$$\text{Fill} = \frac{\mu_{\text{part}} - \mu_{\text{empty}}}{\mu_{\text{filled}} - \mu_{\text{empty}}}.$$

Where $\mu_{\text{part}}$, $\mu_{\text{empty}}$ and $\mu_{\text{filled}}$ stand for the attenuation of the voxel in the volume partly filled with cement, without cement and completely filled with cement respectively. We would like to know the filling percentage for every voxel, but we have only projection images for the case where only some cement is injected. This means we cannot know the position, $S$, of the attenuation coefficients along the ray. Thus we will be comparing the total attenuation of the whole ray

$$\mu(D, E) = \int_0^D \mu(s, E) ds. \quad (8)$$

To get this value from our projection images we can take the natural logarithm to reverse the exponential operation from equation (1).

$$\mu(D, E) = -\ln \frac{I(D)}{I_0} \quad (9)$$
As we will see with equation (12). Expressing the filling 
$I_D$ 
Even if we keep $\ln I_D$ if will not matter for calculating the 
filling percentage as long as $I_0$ is the same for all images. 
As we will see with equation (12). Expressing the filling 
percentage defined by equation (7) in terms of our input 
images $I_D$ _filled_, $I_D$ _empty_ and $I_D$ _part_ gives us 

$$Fill_{perc}(D) = \frac{-\ln I_D(part) + \ln I_D(empty)}{-\ln I_D(filled) + \ln I_D(empty)}$$

We can simplify this to 

$$Fill_{perc}(D) = \frac{\ln(I_D(empty)/I_D(part))}{\ln(I_D(empty)/I_D(filled))}$$

When computing the above a few cases result in undefined 
outcomes, for instance a denominator of a fraction or the 
argument of a logarithm being zero. These cases should be 
taken care of separately.

Now we have the filling percentage of the volume in 
a projection view, shown in figure 8a. To give a better 
overview which areas of the cement target volume are filled 
we would like to provide this information as a volume. 
For a single projection view this is done by simply back 
projecting the projection image over the cement volume 
target. We crop away everything outside the cement target 
volume, because we assume the cement is restricted to this 
volume. An example of such a volume reconstruction is 
shown in figure 9. Having multiple views should make it 
possible to let go of this assumption. However, it will also 
increase the load on the CPU and memory, so it is still 
preferable to restrict the volume of interest as much as 
possible.

Our current implementation only handles DRR images. 
To be able to use real fluoroscopy images we would need a 
few extra steps. First we will need an algorithm to align the 
fluoroscopy images with our CT data. This will enable us 
to make DRRs from the exact same position. Then we will 
need to perform some form of normalization such that the 
intensities in both images correspond to each other. This 
can be done by adjusting the intensities in the fluoroscopy 
image, adjust the DRR generation parameters or both. 
Finally the images need to be of the same resolution such 
that we can compare the images pixel by pixel.

To calculate the total filling percentage one could use 
the reconstructed volume. It is more efficient to use the 
projection image however. Thus we have to weigh the 
values of the projection image by the thickness of the 
cement target volume per ray. This information can be 
obtained by using the DRR ray caster to make a DRR of the cement target volume only. Using equation (8) on 
the cement target volume we measure what part the ray 
travels through the cement target volume. This means we 
can apply the same logic here as we did to the filling 
per ray calculation and take the natural logarithm of 
this DRR image to retrieve the relative weights. With 
$I(D)_cement$ being the projection image of just the cement 
target volume we can calculate the relative weight of a ray 

$$Weight_{ray}(D) = \ln I(D)_cement$$

Then the total filling percentage is 

$$Fill_{perc-total} = \frac{\sum Weight_{ray}(D) \cdot Fill_{perc}(D)}{\sum Weight_{ray}(D)}$$

5.2.1 Uncertainty Calculation

Now that we have calculated how much of the cement 
volume is filled it is also interesting to know how certain 
we are of this filling. For this we can use a measure for uncertainty. This uncertainty indicates how much the 
calculated filling percentage can deviate from the actual 
percentage of the target volume that is filled with cement. 
Because the attenuation of the X-rays has exponential 
behaviour, the intensity values in the fluoroscopy images 
also have an exponential relation to the amount of cement 
present. Thus the amount of cement is not linearly pro-
portional to the intensity values in the fluoroscopy image. 
The result is that the uncertainty of the filling depends 
both on the uncertainty and the intensity of the pixels 
in the fluoroscopy image. The uncertainty is largest when 
pixels in the fluoroscopy image are completely black. When
Figure 9. These figures show the result of the cement filling uncertainty computation, visualized with yellow to blue for the amount of filling and light to dark for the uncertainty. In figure (a) slice 160 of the reconstructed cement filling volume is selected and in figure (b) slice 200 is selected. At the top the input volume giving the filling per pixel is displayed. The light and dark grey parts at the top should correspond to the yellow and blue parts respectively in the reconstructed cement filling displayed at the bottom as a slice view and at the right in a volume visualization. In the reconstruction images some parts are black, indicating that it is impossible to say anything about the amount of filling at those positions. These parts are located either in front or behind the prosthesis, which blocks all radiation.

this happens it is nearly impossible to say anything about the cement filling as we have no way of knowing how much material actually is between the source and the detector. All we know is that there is enough to attenuate all the radiation for the corresponding pixel. So when cement is added to these parts the resulting image will not change intensity and we can thus not detect the cement here. When pixels in the image are completely white it means that there is no material between the source and detector. So when cement is added to these parts we have minimal uncertainty, because all attenuation happens due to the cement and thus the intensity of these pixels is solely determined by it. The benefit of calculating the uncertainty information is that we can inform the user, take it into account when reconstructing our cement filling volume and turn it into a measure for optimizing the viewing angle of the fluoroscopy C-arm.

We can measure the uncertainty of the filling percentage by calculating how our cement filling percentage responds to intensity changes in the projection images. The filling percentage $Fill_{perc}(D)$ has a logarithmic relation to the input images, as can be seen in equation (13).

To calculate the sensitivity of the cement filling to the fluoroscopy input image we can take the derivative of the cement filling (equation (13)) with respect to the fluoroscopy input image $I(D)_{part}$:

$$Fill_{sens}(D) = \frac{\partial}{\partial I(D)_{part}} \ln \frac{I(D)_{empty}}{I(D)_{part}}$$

Which results in

$$Fill_{sens}(D) = -\frac{1}{I(D)_{part} \cdot \ln \frac{I(D)_{empty}}{I(D)_{filled}}}$$

It is possible to find the total uncertainty in the same manner as it is done for the filling percentage. This total uncertainty can be used for example as a measure of how well visible the cement is. We need the weight of the ray, $Weight_{ray}(D)$ as calculated in equation (14) and then use an equation similar to equation (15), but this time with the filling uncertainty, $Fill_{sens}$. The total sensitivity is

$$Fill_{sens-total} = \frac{\sum Weight_{ray}(D) \cdot Fill_{sens}(D)}{\sum Weight_{ray}(D)}$$

5.3 Cement Volume Reconstruction

To give a more detailed view of the situation we can, instead of providing a projection view, reconstruct a volume which indicates what parts are filled with cement. Recovering the cement volume from a single projection image can leave large parts of the area of interest unknown. This happens both because of the prosthesis blocking part of the view and the fact that we have no information about the depth of the cement. To alleviate this problem we can use multiple projection images from different viewing angles to reconstruct the injected cement volume. Reconstructing a volume from a single view does not add any information and interpreting the calculated filling and uncertainty also becomes harder. For projection views it is clear what the filling values mean. The percentage indicates how much of the underlying volume is filled and the uncertainty indicates how accurate this percentage value is. With the data smeared across the volume the percentages indicate the chance of the voxel being filled and the uncertainty indicates how accurate this chance is. There are several options to use for reconstruction. The options we have investigated are: Back Projection [Prince and Links, 2006;
Gonzalez and Woods, 2008] and Algebraic Reconstruction [Kak and Slaney, 2001], which is also used by Tomazevic et al. [2006]. Keep in mind that when adding uncertainties it should be done as a sum of squares. This means it is not possible to just average the uncertainties and extra care has to be taken to correctly accumulate the uncertainty of the voxels.

5.3.1 Back Projection

Back Projection does the following. Every projection image is projected back onto the volume with its corresponding angle so it is consistent with the projection at the same angle. Averaging all back projected volumes yields the final reconstructed volume. This is also the basis for CT reconstruction algorithms. Our data is different from conventional images used with Back Projection. Our data does not contain attenuated values but an average. Additionally each value also has uncertainty. Instead of averaging out all values when back projecting, we back project the values directly and do not average them over the length of the ray. The uncertainty can also be back projected together with the percentage values and this information can then be used when averaging the back projected volumes to obtain the final reconstructed cement volume. This can be done by averaging with weights that are inversely proportional to the uncertainty of the data. This should provide less uncertainty overall and thus more reliable results.

5.3.2 Algebraic Reconstruction

The basis of algebraic reconstruction algorithms is explained by Kak and Slaney [2001]. Algebraic reconstruction is performed by a set of linear equations whose unknowns are elements of the object to be reconstructed. In the conventional matrix equation 19, \( x \) represents our reconstructed volume, \( A \) would contain the weights for the voxels of the reconstructed volume in each ray and \( b \) holds all projection image pixels of all projection images.

\[
A \cdot x = b \tag{19}
\]

The set of linear equations cannot be solved by inversion with conventional matrix theory methods as the matrix would be too large to fit in memory. For one single projection image of 256 \( \times \) 256 pixels \( b \) would be in the order of 65,000 elements and with a volume \((x)\) of similar size the matrix would be 65,000 \( \times \) 65,000. This would result in about 16 Gigabytes of memory when using four byte floating point values. Luckily we can also solve this system of linear equations in an iterative fashion, so we won’t need to store the full matrix \( A \).

We can write a set of linear equations that relates the voxels (\( f \)) in the CT volume to the pixels (\( p \)) in the projection fluoroscopy image

\[
\begin{align*}
  w_{11} f_1 + w_{12} f_2 + \ldots + w_{1N} f_N &= p_1 \tag{20a} \\
  w_{21} f_1 + w_{22} f_2 + \ldots + w_{2N} f_N &= p_2 \tag{20b} \\
  w_{31} f_1 + w_{32} f_2 + \ldots + w_{3N} f_N &= p_3 \tag{20c} \\
  \vdots \\
  w_{M1} f_1 + w_{M2} f_2 + \ldots + w_{MN} f_N &= p_M \tag{20d}
\end{align*}
\]

The weights (\( w \)) indicate the contribution of the voxels for each pixel. Many of these contributions will be zero because there is only a small subset of the volume which contributes to one pixel in the projection image. An image with \( N \) voxels results in \( N \) degrees of freedom. Therefore, an image represented by \((f_1, f_2, \ldots, f_N)\) corresponds to a single point in an \( N \)-dimensional space. In this space equations (20a) to (20d) represent hyperplanes. To locate the solution we start with an initial guess. We then project this initial guess on the first hyperplane, then reproject the resulting point on the second hyperplane and so on for all hyperplanes. When we reach the last hyperplane we reproject the resulting point again on the first hyperplane and so forth until we find the solution. Updating the solution can be done with

\[
\vec{f}^{(i)} = \vec{f}^{(i-1)} - \frac{\vec{f}^{(i-1)} \cdot \vec{w}_i - p_i}{\vec{w}_i \cdot \vec{w}_i} \vec{w}_i. \tag{21}
\]

The advantage of this method is that we could potentially incorporate more specialized reconstruction rules, such as enforcing the cement to be connected (for each entry point).

If a unique solution exists, the iterations will always converge to that point. When we have a overdetermined system no unique solution exists. The result will not converge to a single point, but oscillate between different solutions as mentioned by Kak and Slaney [2001]. When the system is underdetermined we can have an infinite number of solutions. In this case it will converge to a point closest to the initial guess as long as the system has nonzero rows as proven by Tanabe [1971].

As stopping criterion there are several options. First when quality is the main concern we can either use an measure of improvement which compares the current solution to the previous. Second if time is more of an issue a limit can be set on the number of iterations. Third it could be interesting to stop based on a measure of total uncertainty. Last it is of course possible to use multiple stopping criteria. For our experiments we only used a fixed amount of iterations.

In a usual reconstruction from projections the projection images contain accumulated values. In our filling images however we have percentages which are not accumulated, but average values. This means that instead of comparing \( p_i \) to the accumulated value

\[
\vec{f}^{(i-1)} \cdot \vec{w}_i \tag{22}
\]

we need compare

\[
p_i \cdot \sum_j w_{ij} \tag{23}
\]

to the accumulated value. So we need to modify equation (21) to interpret our data correctly. This results in

\[
\vec{f}^{(i)} = \vec{f}^{(i-1)} - \frac{\vec{f}^{(i-1)} \cdot \vec{w}_i - p_i \cdot \sum_j w_{ij}}{\vec{w}_i \cdot \vec{w}_i} \vec{w}_i. \tag{24}
\]

We can also take the uncertainty of the projection values into account by weighting the correction term by one
minus the uncertainty \( (u_i) \) resulting in
\[
\tilde{f}^{(i)} = \tilde{f}^{(i-1)} - (1 - u_i) \cdot \frac{\tilde{f}^{(i-1)} \cdot \vec{w}_i - p_k \cdot \sum_j w_{ij} \cdot \vec{w}_j}{\vec{w}_i \cdot \vec{w}_i} .
\] (25)

This is similar to how relaxation parameters are applied such as in Censor [1983]. The difference is that in our case not all rays are relaxed by the same amount, so some contribute more than others. The visual quality of the solution can be improved by, instead of updating the solution ray by ray only, updating the solution only after we have gone through all equations (called SIRT) as discussed in Kak and Slaney [2001]. In the ART algorithm explained in Kak and Slaney [2001] a binary approximation for the weight \( \vec{w}_i \) is used, but since it’s fairly easy to calculate the weights before iterating through all rays we decided not to use an approximation for the weights. Further improvements are made in the SART algorithm where they use bilinear elements, partial weights and apply a Hamming window on the rays Kak and Slaney [2001]. There exist many variations on the standard Algebraic Reconstruction method that can improve the quality of the reconstruction such as CART (Jonges et al. [1999]) or AART and MART used by Konovalov et al. [2006].

### 5.4 Finding the Optimal Viewing Angle

The information we acquired using the theory in sections 5.1 and 5.2 can be used to determine the best viewing angle of the fluoroscopy C-arm. We can alter the orientation of the C-arm to determine good viewing angles with measures based on inserted needles and cement. The optimization method we used is the Nelder-Mead simplex. We have measures to optimize for needle and cement filling perception. The needle measures take the visibility and sensitivity of a needle into account. The cement filling measures are based on the visibility and uncertainty of the cement.

In our case we consider the angles of the C-arm fluoroscope variables, but neither the distance to the target nor the position, resulting in a two dimensional (2D) search space. The imaging target is the hip prosthesis and its surroundings, which is assumed to be static for the duration of the operation. Searching for an optimal viewing angle in a two dimensional space instead of a six dimensional space simplifies the computation considerably.

The optimal viewing angle can be determined by a brute force method. This means computing the C-arm angle score for every possible angle, which is not efficient. A faster solution would be using a common optimization algorithm like gradient descent, downhill simplex or genetic algorithms. We do not have explicit gradient information and the only way to retrieve it is to approximate it with a discrete difference. That is why we decided to use the downhill simplex algorithm because it does not need explicit gradient information, unlike the gradient descent and related methods by Nelder and Mead [1965]. The downhill simplex algorithm creates a \((n + 1)\)-dimensional simplex for a \(n\)-dimensional search space.

#### 5.4.1 Viewing Angle Scores

To optimize the viewing angle we need to measure the quality of the view for a given angle. The result of this measurement we call the viewing angle score, which is a function of: the angle, needle pose and the cement distribution. The final measure consists of several subparts: needle visibility, needle movement visibility (accuracy), cement visibility and cement uncertainty (Discussed in section 5.2.1).

First of all we create the measure for needle visibility, which considers the area of the image covered by the inserted needle(s). To account for different resolutions we use the relative area of the image rather than the absolute area. The needle visibility score is:
\[
S_n = \frac{\sum \text{I}_n}{\text{I} \cdot \text{n}} .
\] (26)

Here \( \text{I}_n \) is the amount of pixels covered by the needle(s) and \( \text{I} \) the total number of pixels in the projection image.

Secondly we will create a measure for the visibility of needle movement, which we call needle accuracy. This measure arises by applying small permutations, \( \delta \), to the needle pose and analysing the resulting images. This is done by computing the difference images \( (\text{I}_d) \) of the images with the permuted needle positions relative to the neutral position. Needle accuracy is defined as:
\[
S_a = \frac{\sum \text{I}_d}{\sum \text{I}_n} .
\] (27)

Apart from considering needles we also consider the cement target. So thirdly we create a measure for the cement visibility. We do this in a similar manner as the needle visibility measure (equation (26)); we consider the area covered by cement in the projection image \( (\text{I}_c) \) and calculate the relative covered area:
\[
S_c = \frac{\sum \text{I}_c}{\sum \text{I}} .
\] (28)

Lastly we use the uncertainty of the cement filling calculated in section 5.2.1. However for the uncertainty we need to know a filling of the target volume. Since we are interested in completely filling the target volume we use a completely filled target volume to calculate the uncertainty of the cement filling. It is possible to use the total uncertainty calculated by equation (18) as the score \( S_s \).

For different steps of the minimally invasive hip fixation procedure we can use different combinations of sensitivity measures. During needle placement the cement is not relevant and so a combination of needle measures \( S_n \) and \( S_a \) is sufficient. This score is then calculated by:
\[
S_{na} = S_n \cdot S_a \propto \sum \delta \sum \text{I}_d .
\] (29)

After a needle is placed we assume it stays in place. This means that the cement visibly dominates the viewing angle score. The combined cement measure is defined by
multiplying the cement visibility $S_c$ and the uncertainty in the filling of the target $S_s$. This results in:

$$S_{cs} = S_c \cdot S_s \propto \sum I_c \cdot \frac{\sum Weight_{ray}(D) \cdot Fill_{sens}(D)}{\sum Weight_{ray}(D)}.$$  

The search spaces for needle and cement viewing angle optimization are displayed in figures 11b and 11d. Those correspond to the situation depicted in figure 10.

5.5 Needle Sensitivity

To compensate for the loss in 3D information from no longer using a intraoperative CT scanner we investigated the effect of attaching geometric markers to a needle. The purpose of these geometric makers is to help the surgeon determine the orientation of a needle from just the fluoroscopy image. Recognising the orientation will be easier if a needle with the geometric markers produces a different fluoroscopy view for different orientations. Rotations around the axis of the viewing direction are no problem as these show up as rotations in the 2d fluoroscopy view. What is more important to accentuate is the rotations perpendicular to the viewing direction. For these rotations we should note that the incision was not taken into account in the planning, so additional cement went into it during the procedure. Also the needle models we used are slightly larger than the ones used in the cadaver experiment. Overall the DRRs seem to portray the real situation well enough for visual comparison when all elements are in the correct place.

6.1 Digitally Reconstructed Radiographs

The quality of DRRs has been evaluated by visually comparing their likeness to real fluoroscopy images. One difficulty of comparing the DRRs and fluoroscopy images is finding the correct angle and distance from which the fluoroscopy image was recorded. A second difficulty is the fact that the soft tissue will not be in the same position when the patient is moved. So between the recording of the CT volume and the fluoroscopy image the soft tissue will have moved with respect to the bones and prosthesis and thus the DRR image will never be exactly the same as the fluoroscopy image. Real fluoroscopy images often are of higher resolution than the DRRs, because the recorded CT volumes are not of high enough resolution to be able to match the fluoroscopy images. Thus the main purpose of the DRRs is to show the situation of the bone, prosthesis, needle(s) and cement, so this is what we will focus on when comparing the images.

Figure 13 shows fluoroscopy images taken during a cadaver experiment and DRRs made using the corresponding CT dataset. In figures 13b and 13c the situation is not exactly the same because the DRRs are made preoperatively. However they do show the same structures. Figure 13f shows a DRR made post-operatively to compare to the intra-operative in figure 13e. The intraoperative images are slightly lighter and show the incision that was necessary to prepare cement target lesions more clearly than in the DRRs. Figures 13c to 13e and figure 13f also show cement. It is clear that the cement is better visible in the fluoroscopy images than in the DRRs. This is more severe for for preoperative DRR. In the postoperative DRR there are differences in intensity for the cement and around the prosthesis. We should note that the incision was not taken into account in the planning, so additional cement went into it during the procedure. Also the needle models we used are slightly larger than the ones used in the cadaver experiment. Overall the DRRs seem to portray the real situation well enough for visual comparison when all elements are in the correct place.

6.2 Cement Filling Reconstruction

The cement reconstruction was tested on simulation data. We took a real CT scan and added cement in the same way as explained in section 5.1.2 and used DRRs with and without added cement to calculate the filling percentages of the cement. Ideally we would like to propagate the uncertainty of the fluoroscopy projection image to the filling percentage uncertainty. Since our implementation is not able to handle real fluoroscopy data we used simulated data. To simulate uncertainty we used a discrete version of $I(D)_{final}$ and assume the discrete step size is our uncertainty.

One set of results is shown in figure 8. We can see that the part of the image covered by the prosthesis is fully uncertain, because all the radiation gets attenuated by the prosthesis. The other parts filled with cement also have some uncertainty but these levels are so low that they

6 Results

In this section we will investigate the results of the generation of digitally reconstructed radiographs (DRR), cement filling uncertainty calculation, the overlay mode for the DRRs and the viewing angle optimization.
Figure 11. The distribution of the score computed from equation (26) in (a), equation (29) (which is proportional to equation (27)) in (b), equation (28) in (c) and equation (30) in (d). Brighter means a better score. The ten red dotted isolines are drawn at regular intervals. The optimal angles found by brute force and simplex optimization are presented in table 1.

Figure 12. This figure shows the different needle designs that were tested. From the left to the right we have, in (a), the conventional needle, in (b), the circular attachment needle, in (c), the spherical attachment needle, in (d), the triangular attachment needle and finally, in (e), we have the geometric attachment needle. The lettering corresponds to the graphs shown in figure 18.

are hardly visible without increasing the brightness and/or contrast in the image.

In figure 9 the result of figure 8 is back projected into the cement target volume. Here we can also see the cement filling that we simulated. The reconstructed filling approximates the original filling we provided. However the quality of the reconstruction strongly depends on the viewing angle. When there is both a largely filled and unfilled area behind each other the resulting filling percentage will be an average of the two.

We used the Back Projection and Algebraic Reconstruction methods from section 5.3 to reconstruct a cement filling volume from two fluoroscopy projection images, see figure 14. Both methods provide us with more information compared to the single view reconstruction figure 9. The Algebraic Reconstruction method does suffer from some irregularities which in some situations disappear with more iterations as visible in figure 14c. In other situations the irregularities get amplified, such as visible in the top images in figure 14b and figure 14c. The Back Projection method produces smoother results. For the unweighted averaging between Back Projected volumes (Figure 14d) we do get added information, but also existing information has become less certain. This is no longer the case for the weighted averaging (Figure 14e). All reconstructions do however also give incorrect results as is most apparent in the images on the bottom row, which show slice 200 of the
Figure 13. These are fluoroscopy images. The images in (a), (c) and (e) are recorded during a cadaver experiment and the images in (b), (d) and (f) are generated from preoperative CT-data. The fracture in this cadaver leg was created to be able to hollow out the bone to simulate a fibrotic tissue area. Images (c), (d), (e) and (f) are after the fibrotic tissue area is filled with cement, while images (a) and (b) are without any cement.

<table>
<thead>
<tr>
<th>Object</th>
<th>Method</th>
<th>Space</th>
<th>α</th>
<th>γ</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Needle</td>
<td>Simplex</td>
<td>Full</td>
<td>53.75</td>
<td>28.29</td>
<td>9.45510e-4</td>
</tr>
<tr>
<td>Needle</td>
<td>Brute Force</td>
<td>Full</td>
<td>55</td>
<td>25</td>
<td>9.45574e-4</td>
</tr>
<tr>
<td>Cement</td>
<td>Simplex</td>
<td>Full</td>
<td>72.52</td>
<td>20.74</td>
<td>1.30944e-2</td>
</tr>
<tr>
<td>Cement</td>
<td>Brute Force</td>
<td>Full</td>
<td>75</td>
<td>20</td>
<td>1.31231e-2</td>
</tr>
<tr>
<td>Needle</td>
<td>Simplex</td>
<td>Bounded</td>
<td>30.00</td>
<td>30.00</td>
<td>9.02242e-4</td>
</tr>
<tr>
<td>Needle</td>
<td>Brute Force</td>
<td>Bounded</td>
<td>30</td>
<td>30</td>
<td>9.02242e-4</td>
</tr>
<tr>
<td>Cement</td>
<td>Simplex</td>
<td>Bounded</td>
<td>23.51</td>
<td>11.45</td>
<td>1.18845e-2</td>
</tr>
<tr>
<td>Cement</td>
<td>Brute Force</td>
<td>Bounded</td>
<td>-30</td>
<td>05</td>
<td>1.22147e-2</td>
</tr>
</tbody>
</table>

Table 1

Here we have the resulting angles for the fluoroscope C-arm from the simplex optimization and brute force search, both for the bounded and unbounded cases. The brute force search has been performed on a five degree granularity. The simplex optimization was done with an accuracy of 0.01 degrees.

reconstructed volume. This means the filling percentages in the projection image for the second angle provided us with incorrect values.

6.3 Fluoroscopy C-arm Optimization

Optimizing the Fluoroscopy C-arm to get the best view of needle placement and cement injection can lead to results shown in figure 15. The angles presented here are however not feasible in practice, so in order to get usable results we limited the angles of the optimization space. We then obtained the results in figure 16. The results of the different optimizations are presented in table 1. Here we can see that the simplex optimization does not always give us the most optimal angle, such as in the bounded cement case. However the score is within 3% of the score from the optimal solution found by the brute force search. In all other cases presented in this work the simplex optimization produced the result closest to the result of the brute force search. This indicates that even though the simplex optimization might not result in the most optimal fluoroscopy C-arm angle every time it does find a solution that is locally optimal. As figure 11 indicates the search space does not have a lot of local maxima. Thus the simplex optimization is unlikely to get stuck in a local maximum that is significantly worse than the global maximum. Also, usually one will limit the search space to a smaller area, because the fluoroscopy C-arm is not able to move to or record images from all angles during the operation.

6.4 Needle Sensitivity

We tested the needle designs in figure 12 on the relative difference with the anterior view image. This is computed by the following equation where $I_0$ is the front view image and $I_\phi$ is the image with the needle rotated by $\phi$ degrees:

$$D_{rel} = \frac{\sum |I_0 - I_\phi|}{\sum I_0}$$  \hspace{1cm} (31)

We can see in the graphs in figure 18 that for the plain needle the relative difference is lower than for the rest.
Figure 14. These figures show the result of the cement filling reconstruction from two different angles using the Algebraic Reconstruction method from section 5.3.2 and using Back Projection from section 5.3.1, visualized similar to figure 9. On the top row we have slice 160 and on the bottom we have slice 200. From left to right we have: the original filling volume (a), Algebraic Reconstruction after 2 iterations (b), Algebraic Reconstruction after 10 iterations (c), Back Projection with unweighted averaging (d) and Back Projection with weighted averaging (e). The viewing angle for the first fluoroscopy image used in the reconstruction is the same as in figure 9 and the second one is rotated by 30°. We can see that more iterations for the Algebraic Reconstruction method produce better defined results. The Back Projection method with unweighted averaging gives us more information than for reconstruction from a single view, but it comes at a cost as previously certain areas become more uncertain. This is resolved by weighting the values by with the inverse of the uncertainty when averaging the Back Projected volumes.

Figure 15. These figures show the optimal position found by the simplex optimization algorithm for the situation depicted in figure 10. In (a) the optimal position for a needle is shown and in (b) we have the optimal position for the cement. Also shown in table 1.

Figure 16. These figures show the optimal position found by the simplex optimization algorithm for the situation depicted in figure 10, but this time with a bounded search space. In (a) the optimal position for a needle is shown and in (b) we have the optimal position for the cement. Also shown in table 1.
Figure 17. Here we have a few needle designs DRRs at different tilting angles. From left to right the angles are: 30°, 15°, 0°, −15°, −30°. In (a) we have the conventional needle. In (b) we have the circular attachment needle. In (c) we have the spherical attachment needle. In (d) we have the geometrical attachments needle.

Together with the needle with the circular attachment they have less variance with different needle axis rotations. This is not surprising as they are circularly symmetrical around the needles axis. What is surprising is that for the circular attachment needle the difference goes down again after 20°. This could be due to the attachment as a whole getting smaller in the image because it gets turned away from the source.

Being able to tell if a needle is tilted is one thing, but it is not always possible to tell which way a needle is tilted looking at just a projection image. This is the case when a needle has a certain degree of symmetry. This is solved by making a needle asymmetrical, as we have done with the geometrical attachment needle. The difference can be seen in figure 17. For most needles it is nearly impossible to tell if a needle is tilting forwards of backwards, but because every individual attachment on the geometric attachment needle has a different shape we can tell what way a needle tilts by which shape goes up in the image and which one goes down.

For the conventional needle it is difficult to tell how it is tilted in the images, because only the length changes when tilting the needle. The circular attachment needle clearly changes shape in the projection image when tilted. However it is nearly impossible to tell which way the needle is tilted, because the projection image looks similar for both a forwards and backwards tilted needle. The spherical attachment needle also is similar to the circular attachment needle in terms of how well we can tell the tilting of the needle. The spheres are displaced from each other more with different amounts of tilting, but it is hard to tell which way the needle is tilted. The geometrical attachments needle has the potential to solve this problem. Without prior knowledge it is still difficult to tell how the needle is tilted in the images, but because we can deduce how the needle attachments are orientated with respect to the fluoroscope it is possible to retrieve the orientation from the fluoroscopy view. In this case the needle is rotated around its axis by 180° relative to the model in figure 12 and the sphere attachment is thus pointing towards the viewer. For example in figure figure 17d on the left we see the sphere is above the cube and because it is pointing towards us we know the tip of the needle is tilted away from the view. Thus the geometrical attachment needle design has the most potential to provide increased perception of a needles tilting angle. It has three different shaped attachments which make it a needle tilting forwards and backwards look different in the projection view of a fluoroscopy C-arm.

7 Recommendations for Future Work

At its current state the cement filling calculation does not perform any registration of the input images. This means that we are only able to perform the calculations on DRRs made by our current software taken from the same angle.
Figure 18. These are the graphs showing the relative difference in covered image area for the needle designs compared to the neutral position of $0^\circ C$ at various tilting angles. The lettering of the graphs corresponds to the needles shown in figure 12. The conventional needle, shown in (a), clearly performs worst of all. Thus any attachment to a needle will be better than none. It is interesting to see that the circular attachment needle, shown in (b), actually shrinks in screen area when tilting the back (Where the attachment is.) of the needle away far enough. The spherical, triangular and geometric attachment needles, shown in respectively (c), (d) and (e), all show similar performance here. However the geometric attachment needle the advantage, because it's attachments can be distinguished from each other.
To be able to handle intraoperative fluoroscopy images the software will first of all need to be able to retrieve the viewing angle and position of the fluoroscopy C-arm relative to the CT volume. This would enable us the compute DRRs which correspond to the fluoroscopy images. Secondly it would also need to match the brightness and contrast between the fluoroscopy images and DRRs before we can perform the cement filling and uncertainty computation.

Improving the cement filling uncertainty calculation itself is also a possibility. The uncertainty calculation currently uses the the discretization of the generated DRRs. This could be improved upon by using information about the sensitivity, accuracy and/or the signal-to-noise-ratio of the fluoroscope itself.

We have taken the first steps in using multiple viewing angles with a fluoroscopy C-arm to reconstruct injected cement. This can benefit from additional research, especially in how to combine the multivariate data that results from the cement filling computation. Currently the cement filling is given by a percentage, but for a volume representation it would make more sense to have a binary value (yes or no). With only one projection view it is as good as impossible to provide this kind of information. However with multiple projection views taken from different angles it could be possible to reconstruct a binary representation of the cement filling. The uncertainty would then indicate how certain we are of the binary value, which is easier to interpret than a uncertainty of a filling represented by a percentage. Additionally the cement filling uncertainty computation based on multiple projection images (DRRs) can be improved upon by taking into account how the cement may have been able flow. This puts more constraints on the cement distribution and can help in convergence of the cement filling reconstruction. A possible difficulty might be predicting the flow if the physical situation changes during the procedure such as the bone getting damaged during needle insertion. However even if not taking into account the surrounding tissue it should already help to constrain the cement to a limited amount of blobs. The cement is injected only in a few spots and thus we will only have a known number of bodies of cement. It is also unlikely that the cement will fill voxels for a fraction when they’re not on the edge of a cement blob. This means that in all non edge cases the cement either fills a voxel completely or not at all.

8 Conclusion

At the LUMC a novel method for refixating loosened hip prostheses has been developed. This currently uses a intraoperative CT-scanner to guide the operation. This means that the requirement to perform the procedure is a CT-room suitable for surgical operations, which is not widely available. To ensure the consistency of this method and make it verifiable and better accessible to general hospitals we developed a planning system and verification tools. We focussed on using just a fluoroscope for intraoperative guidance (next to the preoperative CT scan) and providing tools to verify the results based on fluoroscopy images. To make up for the lost 3D information provided by the CT scanner during needle insertion we suggested improving needle sensitivity by attaching asymmetric objects to a needle. We took the first steps with the planning system and verification tools, making it possible to simulate fluoroscopy images of intraoperative situations and providing a new method which has the potential to give insight into the cement filling from intraoperative fluoroscopy images and a preoperative CT scan. In its current state the verification tool does not perform any registration of the images. This means that we are only able to perform the calculations on DRRs made by our current software taken from the same angle.

The provided planning and verification tools include a 3D view of the CT data, DRRs, C-arm viewing angle optimization, cement filling reconstruction. Additionally the provided needle designs look promising for increasing needle sensitivity. This can increase the reliability of needle placement. The DRRs include simulation of the insertion needles and cement in the preoperative CT volume and overlays to highlight the different tissues in a similar manner as provided in the 3D view used to plan needle placement. This enables us to simulate the various steps in the refixation procedure and thus provide the surgeon with a planning which will improve the reliability and consistency of the procedures. The DRRs have been compared to real fluoroscopy images and were found to represent the rigid structures of the objects well. In some areas the DRRs differ in intensity and contrast, but this does not provide a significant obstacle for visual comparison. So the DRRs are very well usable for planning purposes and to act as a guide during an operation.

The verification tool can calculate the amount of cement injected without requiring a postoperative CT volume. Currently only DRRs can be used to compute the cement filling and uncertainty. To enable the use of real fluoroscopy images a registration system and intensity equalization is needed. However, verifying the cement injection will enable the results of the refixation procedure to be investigated at a low cost, which has the potential to increase the success of future implementations.

The data from our needle design simulations shows that attaching objects to a needle has the potential to increase the visual perception of its position. All designs make needle movements more apparent in a fluoroscopy image. However, with many of the designs it is difficult see if a needle is tilted forwards of backwards. This is because the attached objects have a similar shape and are thus indistinguishable. We solved this with our final design: the geometrical attachments needle (Figure 12e). In this design we attached three different geometrical objects to a needle. Because the objects all have a different shape it is possible to determine needle orientation by identifying the relative position of the objects in the fluoroscopy image.

The proposed planning system has the potential to help the surgeon decide on how to monitor the operation using just a fluoroscope. Together with the verification tools it
could be possible to eliminate the need for an intraoperative CT-scanner while still providing the possibility of postoperative verification and evaluation of the injected cement. When refined this will make the minimally invasive hip refixation procedure more accessible and can prolong the lifetime of hip prostheses.

Acknowledgment

This research is conducted in context of a master thesis for the BioMedical Engineering master study at the Delft University of Technology. The author likes to thank the following people for their cooperation and support:

- Charl Botha
- Elmar Eisemann
- Markus Hugot
- Dion Koeze
- Francois Malan
- Marijn van Nijhuis
- Renata Raidou
References


Appendix A
Needle Sensitivity Experiment

The needle sensitivity analysis experiment was performed with the help of the Digitally Reconstructed Radiographs (DRR) generator. Different needle designs were imported and inserted into an empty data volume such that they can be rendered by the DRR generator, which uses volume data with hounsfield units to generate either an X-ray or fluoroscopy image. The mathematics behind the DRR are explained in the main document in section 5.1. We started out improving the standard needle by adding a circular disk. The idea was that this would help in determining the tilting angle of the needle in the projection view. The result was the circular needle design which is shown in figure 12b in the main document. This is obviously better than the conventional needle, but still has problems with regard to determining the direction of the tilting. To improve on this we thought it would be better to use a less symmetric design, which would also be smaller and make the needle easier to handle. The following designs were the spherical attachment needle (figure 12c) and the triangular attachment needle (figure 12d). The spherical attachments did not improve the situation much, so using shapes that were less uniform in shape were thought to be a good improvement. Unfortunately the triangles did not do the job. If we would be able to distinguish the different attachments from each other it would be possible how each attachment moves and would provide us with more information, because we can deduce the direction of the tilting angle. This is true as long as the view of the attachments is not obstructed. In figure 18 the results are shown for the relative difference in viewing area occupied by the needle. From this it’s clear that all attachments have an easier to distinguish projection image. In figure 19 we show the shape number of the needles at different angles. This is calculated by dividing the total length of the outline by the area of the projection shape. It was thought to be a nice way to show the difference in shape, but it does not work very well for most needles. This is because the attachments generally don’t change much in shape, but only differ in their position to each other in the projection image. The most significant parts code with which this experiment was performed is given in listings 1 to 3.
Figure 19. These are the graphs showing the shape number for the needle designs at various tilting angles. The lettering of the graphs corresponds to the needles shown in figure 12 and figure 18. These graphs don’t show much coherent information and were thus not included in the paper.
def TestNeedle(self, extra = ""):
    # reset the camera
    self._handler_reset_camX( None )

    # get the filename and extension and open file to write the data to
    filename, ext = os.path.splitext( self.filename )
    text_file = open( self._config.last_used_dir + "/"+filename+extra".txt", "w"")

    # perform measurements for different angels
    for y in xrange(0,360,30):
        # move the volume instead of the camera (easier)
        self.SetVolumeOrientation(0,y,0)
        # re-adjust view without moving the camera
        self.renX.ResetCameraClippingRange()
        self.render()
        image0 = self.return_render_window_capture()
        y = y%360
    for x in xrange(-30,31,5):
        x = x%360
        self.SetVolumeOrientation(x,y,0)
        self.renX.ResetCameraClippingRange()
        self.render()
        image = self.return_render_window_capture()
        image_diff = self.compute_difference_image( image0, image )
        diff = self.compute_relative_difference_value( image_diff, image0 )
        image_diff = self.compute_relative_difference_value( image_diff, image0 )
        image_edge = self.edge_detection( image )
        # transform to 24bitRGB
        cast = vtk.vtkImageShiftScale()
        cast.SetInput(image_edge)
        cast.SetShift(0.0)
        cast.SetScale(255.0/image_edge.GetScalarRange()[1])
        cast.SetOutputScalarTypeToUnsignedChar()
        cast.Update()
        self.SaveImage( cast.GetOutput(), filename+extra+str(y)+","+str(x)+"diff"")
        self.SaveImage( cast.GetOutput(), filename+extra+str(y)+","+str(x)+"edge"")
        self.shape_number_from_edge(image, image_edge)
        text_file.write( str(y)+","+str(x)+": "+
        str(diff)+"; "+str(shape)+"\n" )
    text_file.close()

    self.make_graph( self._config.last_used_dir + "/"+filename+extra )
def shape_number_from_edge(self, image, edge):
    np_edge = numpy_support.vtk_to_numpy(edge.GetPointData().GetScalars())
    np_full = numpy_support.vtk_to_numpy(image.GetPointData().GetScalars())
    edge_total = (float)(np_edge > (edge.GetScalarRange()[1]/2.0)).sum()
    area = (float)(np_full < (image.GetScalarRange()[1]/2.0)).sum()
    return edge_total / area

Listing 2: This Python code shows how the shape number is calculated. The input is an image of the object and an image containing the edges of the object.

def edge_detection(self, image):
    extractr = vtk.vtkImageExtractComponents()
    extractr.SetComponents(0)
    extractr.SetInput(image)
    extractr.Update()

    sobel = vtk.vtkImageSobel2D()
    sobel.SetInput(extractr.GetOutput())
    sobel.Update()

    extractx = vtk.vtkImageExtractComponents()
    extractx.SetComponents(0)
    extractx.SetInput(sobel.GetOutput())
    extractx.Update()

    extracty = vtk.vtkImageExtractComponents()
    extracty.SetComponents(1)
    extracty.SetInput(sobel.GetOutput())
    extracty.Update()

    square_x = vtk.vtkImageMathematics()
    square_x.SetInput1(extractx.GetOutput())
    square_x.SetOperationToSquare()
    square_x.Update()

    square_y = vtk.vtkImageMathematics()
    square_y.SetInput1(extracty.GetOutput())
    square_y.SetOperationToSquare()
    square_y.Update()

    add = vtk.vtkImageMathematics()
    add.SetInput1(square_x.GetOutput())
    add.SetInput2(square_y.GetOutput())
    add.SetOperationToAdd()
    add.Update()

    sqrt = vtk.vtkImageMathematics()
    sqrt.SetInput1(add.GetOutput())
    sqrt.SetOperationToSquareRoot()
    sqrt.Update()

    return sqrt.GetOutput()

Listing 3: This Python code shows how the edges are detected with the help of the VTK library.
Appendix B
Cement Filling Code

Our implementation of the cement filling calculation from section 5.2 in the main document is described here. First we'll discuss the actual calculation of the cement filling and then we'll discuss the uncertainty and how we acquire all required data.

The calculation of the cement filling in principle is just applying equation (13), however the logarithmic operations and division make for quite a few cases which will result in invalid values for this equation. Thus we have to catch these situations first. The denominators may not be zero and neither can the arguments of the logarithms, so we have to check all if any of the $I(D)$s is zero and we have to decide what filling percentage to assign in each case depending on what values the other $I(D)$s have. This process is shown in listing 4. However the implementation can be made more efficient by making use of the numpy library resulting in the code shown in listing 5.

```python
def calculate_filling(final_image, pre_op_image, full_image, unknown):
    fill_image = numpy.array([0.0]*final_image.size)
    for i in xrange(final_image.size):
        if full_image[i] == 1:
            #not interested in the filling percentage
            #when there is nothing in the volume
            pass
        elif pre_op_image[i] == 0:
            #we cannot detect if anything was filled or not,
            #be conservative
            fill_image[i] = unknown
        elif (full_image[i] == pre_op_image[i]):
            #this breaks the denominator and
            #means there has been no filling
            pass
        elif full_image[i] == 0:
            #we cannot know how much cement should
            #be here as it is fully saturated
            #we already captured the pre_op_image == 0 case,
            #so if it gets here we know it is at least
            #partially filled
            if final_image[i] == 0:
                #it could be fully filled or not...
                fill_image[i] = unknown
            else:
                #it is only partially filled
                fill_image[i] = unknown
        elif final_image[i] == 0:
            #it is either fully filled or more than that
            fill_image[i] = 1  #we do not consider overfilling.
        else:
            fill_image[i] = \n                numpy.log(final_image[i]/pre_op_image[i]) / \n                numpy.log(full_image[i]/pre_op_image[i])
    return fill_image
```

Listing 4: The less efficient Python code variant of the cement filling calculation code in listing 5 that calculates the filling percentage from a projection images of the final situation, preoperative situation, the CT-volume with a fully filled cement target and the value to fill in when the filing is unknown. (Used for calculating the uncertainty later on.) This code is not actually used and is replaced with the code in listing 5.

To calculate the uncertainty we use equation (16). Here the same applies as for the calculation of the cement filling. We have to take into account several special cases and only apply our equation with the logarithms on the pixels which will allow for it. For all the cases where we can’t calculate the uncertainty we set it to full uncertainty, which corresponds to a value of 1.0. The code is shown in listing 6.

Our preparation code for calculating the cement filling and uncertainty is shown in listing 8. We take an image where some cement is inserted and then acquire the following images: An image without any filling, an image with full filling of the cement target and the value to fill in when the filing is unknown. (Used for calculating the uncertainty later on.) This code is not actually used and is replaced with the code in listing 5.
def calculate_filling(final_image, pre_op_image, full_image, unknown):
    fill_image = numpy.array([-1.0]*final_image.size)
    mask_unknown = (pre_op_image==0) + (full_image==0)
    fill_image[mask_unknown] = unknown
    mask_zero = (full_image==1) + (full_image == pre_op_image) * (mask_unknown!=True)
    mask_full = (full_image==0)
    fill_image[mask_full] = 1.0
    mask_log = mask_full + (mask_zero + mask_unknown) != True
    fill_image[mask_log] = numpy.log(final_image[mask_log]/pre_op_image[mask_log])/numpy.log(full_image[mask_log]/pre_op_image[mask_log])
    return fill_image

Listing 5: Python code helper function for listing 8 to calculate the filling percentage. This is a more efficient implementation (and maybe harder to read) of the same algorithm as shown in listing 4.

def calculate_uncertainty(final_image, pre_op_image, full_image, input_uncertainty):
    uncertainty_image = numpy.array([-1.0]*final_image.size)
    mask_unknown = (pre_op_image==0) + (full_image==0)
    mask_zero = (full_image==1) + (full_image == pre_op_image) * (mask_unknown!=True)
    mask_full = (full_image==0)
    uncertainty_image[ (mask_full * (mask_zero !=True) ) + mask_unknown ] = 1.0
    uncertainty_image[mask_log] = -input_uncertainty/ (numpy.log(full_image[mask_log]/pre_op_image[mask_log]) * final_image[mask_log] )
    return uncertainty_image

Listing 6: Python code helper function for listing 8 to uncertainty as per equation (16).

def calculate_weights(cement_only_image, unknown):
    projection_weights = numpy.array([0.0]*cement_only_image.size)
    mask_zero = cement_only_image == 1
    mask_unknown = cement_only_image == 0
    projection_weights[mask_zero] = unknown
    projection_weights[mask_log] = numpy.log(cement_only_image[mask_log])
    return projection_weights

Listing 7: Python code helper function for listing 8 to weights of the pixels in the projection volume so we can determine the total filling percentage as in equation (15).
def calculate_cement_filling(self, cement_fill, azimuth=0, elevation=0):
    self.update_camera(azimuth, elevation)
    # I'm assuming we use grayscale renders, so r=g=b and we can just take one channel.
    self.reset_DRR_volume()
    self.render()
    no_fill_image = \
        numpy_support.vtk_to_numpy(self.return_render_window_capture_float_array())[:,1]
    self.render()
    full_fill_image = \
        numpy_support.vtk_to_numpy(self.return_render_window_capture_float_array())[:,1]

    step = 1.0/cement_fill.GetScalarTypeMax()
    final_fill_image = numpy_support.vtk_to_numpy(cement_fill.GetPointData().GetScalars())[:,0] \
        / float(cement_fill.GetScalarTypeMax())

    # make a zero filled image the same size as self.data
    zero = vtk.vtkImageThreshold()
    zero.SetInput(self.data)
    zero.SetInValue(-100)
    zero.Update()
    # pretend our CT-volume is empty
    real_data = self.data
    zero_data = zero.GetOutput()
    self.data = zero_data
    # render with the cement inserted in the empty volume
    self.update_DRR_volume([],self._cement_mask)
    self.render()
    self.cement_only_image_char = self.return_render_window_capture()
    cement_only_image = \
        numpy_support.vtk_to_numpy(self.return_render_window_capture_float_array())[:,1]
    # restore the CT-volume
    self.data = real_data
    self.update_DRR_volume([],[])

    projection_fill = calculate_filling(final_fill_image, no_fill_image, full_fill_image, 0.5)
    projection_uncertainty = \
        calculate_uncertainty(final_fill_image, no_fill_image, full_fill_image, 1.0/256.0)
    projection_weights = calculate_weights(cement_only_image, 1.0 )

    return projection_fill, projection_uncertainty, projection_weights

Listing 8: Python code for the main body of the cement filling percentage and uncertainty calculation. This code does the preparational work, which consists of acquiring all the images needed. The actual calculations are performed in the helper functions in listings 5 to 7.
Appendix C
Volume Reconstruction Code
Here we provide the code used to implement the different volume reconstruction algorithms described in section 5.3. Our implementation of the back projection algorithm for one projection image is shown in listing 9, for two images using averaging in listing 10 and using weighted sum per voxel in listing 11. The helper function in listing 12 calculates an interpolated value from 4 pixels and the function in listing 13 calculates the indices and fraction which indicates how close the calculated point is to the pixel.

```python
def projection_to_volume(self, projection_numpy_array, volume_mask, azimuth=0, elevation=0):
    self.update_camera(azimuth, elevation)
    render_size = numpy.array(self.renderer.GetSize())
    volume_size = numpy.array(volume_mask.GetDimensions())
    projection_numpy_array = \
        projection_numpy_array.reshape(render_size[1], render_size[0]).swapaxes(0, 1)
    volume_fill = numpy.array([0.0] * volume_size.prod())
    volume_fill_shaped = volume_fill.reshape(volume_size)
    cement_mask = numpy_support.vtk_to_numpy(volume_mask.GetPointData().GetScalars())
    value = numpy.array([0.0] * 4).reshape(2, 2)
    for i in numpy.array(range(volume_size.prod()))[(cement_mask > 0)]:
        pos2d = numpy.array(self.pos3d_to_2d(self.data.GetPoint(i))[:2])  # coordinates are -1->1
        pos2d += 1  # now it is 0 ... 2
        pos2d /= 2  # and now we have 0 ... 1
        pos2d += render_size  # and now it should be 0 ... render_size
        frac = pos2d - pos2d.astype(int)
        volume_fill[i] = calculate_value(projection_numpy_array, frac, i)
    return volume_fill_shaped
```

Listing 9: The Python code that back projects on projection image on a volume.

```python
def projections_to_volume_avg(self, projection_numpy_array1, projection_numpy_array2, \
    projection_numpy_array_unc1, projection_numpy_array_unc2, volume_mask, \
    azimuth1, elevation1, azimuth2, elevation2):
    values=(self.projection_to_volume(projection_numpy_array1, volume_mask, azimuth1, elevation1) \
        + self.projection_to_volume(projection_numpy_array2, volume_mask, azimuth2, elevation2)) / 2.0
    uncertainty = numpy.sqrt(\n        self.projection_to_volume(projection_numpy_array_unc1, volume_mask, azimuth1, elevation1)**2 + \n        self.projection_to_volume(projection_numpy_array_unc2, volume_mask, azimuth2, elevation2)**2 ) / 2.0
    return values, uncertainty
```

Listing 10: The Python code that combines two projection images and takes the average of both to produce one volume.

In listing 14 is our implementation of the algebraic reconstruction algorithm. It uses the helper function in listing 15 which does something similar to the code in listing 13 except that it takes more data points. We first find the area of the projection images that is covered by the cement target volume. Then for each unique position in the projection images we update the volumes containing the cement filling percentage and uncertainty.
def projections_to_volume_best_of(self, projection_numpy_array1, projection_numpy_array2,
projection_numpy_array_unc1, projection_numpy_array_unc2, volume_mask, 
azimuth1, elevation1, azimuth2, elevation2):
    render_size = numpy.array((self.renderer.GetSize()))
    volume_size = numpy.array(volume_mask.GetDimensions())
    projection_numpy_array1 = projection_numpy_array1.reshape(render_size[1],render_size[0]).swapaxes(0,1)
    projection_numpy_array2 = projection_numpy_array2.reshape(render_size[1],render_size[0]).swapaxes(0,1)
    projection_numpy_array_unc1 = projection_numpy_array_unc1.reshape(render_size[1],render_size[0]).swapaxes(0,1)
    projection_numpy_array_unc2 = projection_numpy_array_unc2.reshape(render_size[1],render_size[0]).swapaxes(0,1)
    volume_fill = numpy.array([0.0]*volume_size.prod())
    volume_fill_shaped = volume_fill.reshape(volume_size)
    volume_unc = numpy.array([0.0]*volume_size.prod())
    volume_unc_shaped = volume_unc.reshape(volume_size)
    cement_mask = numpy_support.vtk_to_numpy(volume_mask.GetPointData().GetScalars())
    volume_indices = numpy.array(range(volume_size.prod()))[cement_mask>0]

    self.update_camera(azimuth1,elevation1)
    pos2d1, frac1 = make_pos2d_and_frac(volume_indices)
    self.update_camera(azimuth2,elevation2)
    pos2d2, frac2 = make_pos2d_and_frac(volume_indices)

    for i in xrange(volume_indices.size):
        unc1 = calculate_value(projection_numpy_array_unc1, frac1, i)
        unc2 = calculate_value(projection_numpy_array_unc2, frac2, i)
        fill1 = calculate_value(projection_numpy_array1, frac1, i)
        fill2 = calculate_value(projection_numpy_array2, frac2, i)
        weight1 = 1.0 - unc1
        weight2 = 1.0 - unc2
        tot_weight = weight1+weight2
        if( tot_weight == 0 ):
            weight1 = weight2 = 1.0
            tot_weight = 2.0
        volume_unc[volume_indices[i]] = numpy.sqrt( ( unc1 * weight1 ) ** 2 + ( unc2 * weight2 ) ** 2 ) / tot_weight

        volume_fill[volume_indices[i]] = ( fill1 * weight1 + fill2 * weight2 ) / tot_weight

    return volume_fill_shaped, volume_unc_shaped

Listing 11: This Python code takes two projection images and transforms them into volume using the uncertainty data to weight the values.
def calculate_value(array, frac, i):
    value = numpy.array([0.0]*4).reshape(2,2)
    value[0,0] = array[int(pos2d1[i,0]),int(pos2d1[i,1])]
    value[1,0] = array[int(pos2d1[i,0])+1,int(pos2d1[i,1])]
    value[0,1] = array[int(pos2d1[i,0]),int(pos2d1[i,1])+1]
    value[1,1] = array[int(pos2d1[i,0])+1,int(pos2d1[i,1])+1]
    value[0,0] *= (1.0 - frac[i,0])
    value[1,0] *= frac[i,0]
    value[0,1] *= (1.0 - frac[i,1])
    value[1,1] *= frac[i,1]
    return value.sum()

Listing 12: Python code helper function for listing 11 to calculate one value from four datapoints.

def make_pos2d_and_frac(volume_indices):
    pos2d = numpy.array([0]*volume_indices.size*2).reshape(volume_indices.size,2)
    frac = numpy.array([0.0]*volume_indices.size*2).reshape(volume_indices.size,2)
    for i in xrange(volume_indices.size):
        pos = numpy.array(self.pos3d_to_2d(self.data.GetPoint(volume_indices[i]))[:2])
        #coordinates are -1 ... 1
        pos+= 1  # now it is 0 ... 2
        pos/= 2  # and now we have 0 ... 1
        pos*= render_size  # and now it should be 0 ... render_size
        pos2d[i,0] = pos.astype(int)[:2]
        frac[i,0] = pos - pos2d[i,0]
    return pos2d, frac

Listing 13: Python code helper function for listing 11 to calculate the 2d position and fractional distance to the datapoints.
def projections_to_volume_ART(self, projection_numpy_array1, projection_numpy_array2, 
projection_numpy_array_unc1, projection_numpy_array_unc2, volume_mask, azimuth1, 
elevation1, azimuth2, elevation2, iterations = 2):
    render_size = numpy.array(self.renderer.GetSize())
    volume_size = numpy.array(volume_mask.GetDimensions())
    projection_numpy_array = numpy.append(projection_numpy_array1, projection_numpy_array2)
    projection_numpy_array_unc = \\n        numpy.append(projection_numpy_array_unc1, projection_numpy_array_unc2)
    cement_mask = numpy_support.vtk_to_numpy(volume_mask.GetPointData().GetScalars())
    volume_indices = cement_mask.nonzero()[0]
    pos2d, weights = \
        make_pos2d_and_weight(self, volume_indices, azimuth1, elevation1, azimuth2, elevation2)
    unique_pos = numpy.unique( pos2d )
    unique_pos = numpy.append( unique_pos[0:-1:2], unique_pos[1:-1:2] )
    volume_masked = numpy.zeros(volume_indices.size)
    volume_masked[:] = 0.5
    volume_masked_unc = numpy.ones(volume_indices.size)
    for x in xrange(iterations):
        volume_masked_prev = volume_masked.copy()
        for pos in unique_pos:
            used_voxel_indices = ( pos==pos2d ).nonzero()
            weights = weight[used_voxel_indices]
            used_voxel_indices = used_voxel_indices[0] % volume_indices.size
            percent_filled = projection_numpy_array[pos]
            projection_unc = projection_numpy_array_unc[pos]
            certainty = ( 1.0 - projection_unc )
            weights_sum = weights.sum()
            if( certainty > 0 and weights_sum > 0 ):
                certainty_weights = certainty * weights
                difference = percent_filled - \\n                    numpy.dot( volume_masked_prev[used_voxel_indices], weights )
                volume_masked[used_voxel_indices] += ( ( difference * weights_sum ) / \\n                    numpy.dot( weights, weights ) ) * certainty_weights
                volume_masked_unc[used_voxel_indices] = numpy.sqrt( ( projection_unc * \
                    certainty_weights )**2 + volume_masked_unc[used_voxel_indices]**2 ) / \
                    ( certainty_weights+1)
        numpy.clip( volume_masked, 0.0, 1.0, out = volume_masked )
        numpy.clip( volume_masked_unc, 0.0, 1.0, out = volume_masked_unc )

    volume_fill = numpy.zeros(volume_size.prod())
    volume_fill[:] = -1.0
    volume_fill_shaped = volume_fill.reshape(volume_size)
    volume_unc = numpy.zeros(volume_size.prod())
    volume_unc[:] = -1.0
    volume_unc_shaped = volume_unc.reshape(volume_size)
    volume_fill[volume_indices] = volume_masked
    volume_unc[volume_indices] = volume_masked_unc
    return volume_fill_shaped, volume_unc_shaped

Listing 14: The Python code for our implementation of the algebraic reconstruction algorithm.
```python
def make_pos2d_and_weight(self, volume_indices, azimuth1, elevation1, azimuth2, elevation2):
    pos2d = numpy.zeros((volume_indices.size*2,4,2))
    self.update_camera(azimuth1, elevation1)
    for i in range(volume_indices.size):
        pos2d[i,:,:] = \
        numpy.array(self.pos3d_to_2d(self.data.GetPoint(volume_indices[i]))[:,:])
    self.update_camera(azimuth2, elevation2)
    for i in range(volume_indices.size):
        pos2d[i+volume_indices.size,:,:] = \
        numpy.array(self.pos3d_to_2d(self.data.GetPoint(volume_indices[i]))[:,:])
    pos2d += 1
    pos2d /= 2
    pos2d *= render_size
    pos2d[:,[1,3],0] += 1
    pos2d[:,[2,3],1] += 1
    weight = pos2d
    pos2d = pos2d.astype(int)
    weight -= pos2d
    weight[:,[0,2],0] = 1 - weight[:,[0,2],0]
    weight[:,[0,1],1] = 1 - weight[:,[0,1],1]
    weight = weight.prod(2) # here the size changes to volume_indices.size,4
    pos2d[:,,:] *= render_size[0]
    pos2d = pos2d.sum(2)
    numpy.clip( pos2d, 0, projection_numpy_array.size/2 , out = pos2d )
    pos2d[pos2d.size/2:] += render_size.prod()
    return pos2d, weight
```

Listing 15: Helper function for listing 14 to calculate the pixel indices of the projection image belonging to the voxels in the cement target volume and the weights for those pixels.