

CARLETON UNIVERSITY
PHYSICS DEPARTMENT

An Assessment of a 4D Cone Beam CT System

Author:

Waqar MUHAMMAD

Supervisor:

Dr. E. ALI

Dr. E. HEATH

Dr. T. XU



April 26, 2018

Abstract

Four-dimensional, cone-beam CT (4D CBCT) substantially reduces motion blurring artifacts introduced by thorax motion caused by the respiration in three-dimension CBCT (3D CBCT). However 4D CBCT degrades considerably the image quality compared to a 3D CBCT. In this study we investigate the image acquisition parameters, namely, collimation (small vs medium Field of View (FOV)), and filtration. We conclude that the best image quality, within a reasonable acquisition time and number of projections, was with the medium FOV “Fast” acquisition (1218 Projections, 3.7 minutes). In this time frame it provides a reasonably good image for the purposes of patient positioning. To allow for off-line reconstruction and analysis of these images, we built a 4D CBCT reconstruction process using RTK and compared it to the clinical XVI system it.

Acknowledgements

I would first like to thank my advisors Dr. Emily Heath, Dr. Elsayed Ali, Dr. Tong Xu of Carleton University for their help and guidance throughout this work. Additionally, I would like to thank Eric Christiansen of Carleton University for answering my countless questions. This work would not have been possible without the department of medical physics at Ottawa Hospital Cancer Centre and the RTK Consortium. Lastly, I'd like to thank Carleton University's Physics department for equipping me, over the last four years, with the tools necessary for this undertaking.

1 Objective

The objective of this study is to assess the performance of the Elekta 4D Cone beam CT imaging system at the Ottawa Hospital Cancer Centre, as well as optimize 4D image acquisition parameters for the best image quality. The ability to reconstruct 4D CBCT images off-line is critical to this work, another objective is to build a framework for 4D CBCT reconstruction off-line.

2 Introduction

2.1 4D Cone Beam Computed Tomography

Cone beam computed tomography (CBCT) is a method of imaging where x-rays are projected through a subject onto a flat 2-dimensional detector. The detector consists of a fluorescing material and a CCD capturing device. This method images a 2D projection of the subject as opposed to the 1D slice that would be obtained from a traditional fan-beam CT Scan. Whereas Fan beam CT scans are helical slices that require that the CT Scanner be rotating very quickly and the couch move, a CBCT scan can be performed on a stationary couch, allowing the x-ray source and detector to rotate much slower. This makes it possible to attach a CBCT scanner to Linear accelerators for keV energy level imaging as it is difficult to rotate heavy Linear accelerator heads to the speeds required for Fan Beam CT Scans (typically ranging from 1s/rot - 0.28s/rot) [1]. Four-dimensional CBCT considerably reduces motion blur caused by the respiration motion of the chest in an image that would otherwise be present in a three-dimensional CBCT scan[2], 4D CBCT can be used in free-breathing radiation therapy.

When a 4D CBCT on the chest is taken, the scan's individual projections are then binned into ten phases in the respiratory cycle; a periodic movement that may be described by the cosine power to 4 model [3]. The binning may be achieved by the Amsterdam Shroud method [4]. Each of those bins is then reconstructed using the FDK Algorithm[5] also known as the filtered back-projection scheme. The 3D reconstruction of each of the binned phases result in the reconstructed 4D CBCT, yielding a 3D image at the different position of the tumor throughout the respiratory cycle. This allows the tumor position to be compared to the 4D TPCT.

3 Material and methods

3.1 Feldkamp-Davis-Kress method

Feldkamp-Davis-Kress method (FDK) is an extension of the Filtered back-projection method to cone-beam geometry also known as the Radon Transform used in the conventional Fan Beam CT.

Here we first show the workings of the Radon transform for 2D in the form of a convolution and then projection. When imaging an object made up of different materials, each of which have different attenuation coefficient (μ), x-rays (I) are sent from a source parallel to a detector, such that the x-ray passes through the object and lands on the detector (the resulting beam intensity is I_0). Each x-ray beam attenuates a certain amount depending on how much energy was aborted by the object. As shown in figure 1

$$I = I_0 \exp(-\int_{path} \mu(x) dx) \Rightarrow \int_{path} \mu(x) dx = -\ln(\frac{I}{I_0}) \quad (1)$$

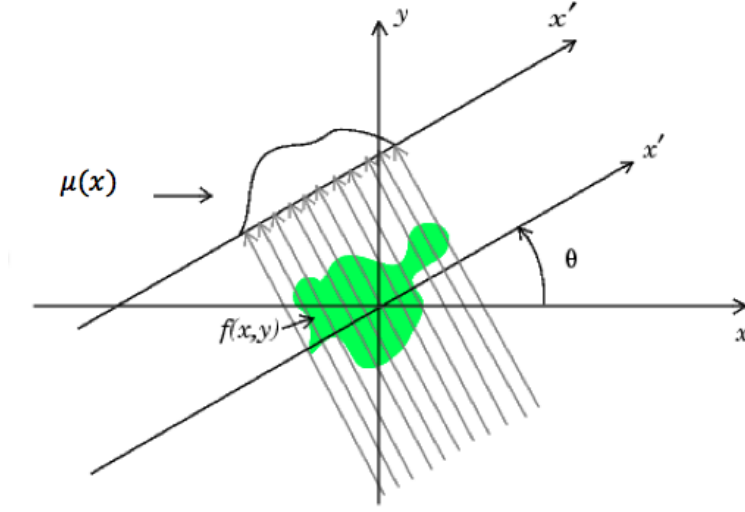


Figure 1: visualization of radon transform

The Radon transform of an image represented by the function $\mu(x, y)$ can be defined as a series of line integrals through $\mu(x, y)$ at different offsets from the origin. Which is mathematically defined as follows:

$$R(r, \theta) = \iint_{-\infty}^{\infty} \mu(x, y) \delta(x \cos \theta + y \sin \theta - r) dx dy [5] \quad (2)$$

Where θ is the angle of the line, and r is the perpendicular offset of the line.

The resulting Radon transform is called a sinogram. In order to reconstruct the image from the sinogram, the inverse Radon transform is to sinogram. There are several methods to compute the inverse Radon transform but the one of interest in this study is the Filtered Back Projection method. This method is split into two phases, the projection and the filtration on the image. The projection phase is very similar to the Radon transform described above, except now the path integral are projected back onto a plane at their respective angles. This back-projection phase takes the form:

$$f(x, y) = \int_0^\pi R(x \cos(\theta) + y \sin(\theta), \theta) d\theta \quad (3)$$

Although the shape of the object is reconstructed, the reconstructed image is very noisy. To correct for this noise a high pass and Hanning filter is applied to the sinogram data in the frequency domain. Applying these filters significantly improves the quality of the reconstructed image.

The FDK Methods extends the Radon transform to conical projections, and the transform can be given by the following: [6]

$$f_{FDK}(x, y, z) = \int_0^{2\pi} \frac{R^2}{U(x, y, \beta)^2} \mu^F(\beta, a(x, y, \beta), b(x, y, z, \beta)) d\beta \quad (4)$$

The detector position is given by:

$$a(x, y, \beta) = R \frac{-x \sin \beta + y \cos \beta}{r + x \cos \beta + y \sin \beta} \quad (5)$$

$b(x, y, z, \beta)$ is given by:

$$b(x, y, z, \beta) = z \frac{\beta}{R + x \cos \beta + y \sin \beta} \quad (6)$$

The function $U(x, y, \beta)$ is the function of the distance between the source and the line parallel with the detector and is given by:

$$U(x, y, \beta) = R + x \cos \beta + y \sin \beta \quad (7)$$

A visualization of the FDK setup can be seen in figure 2

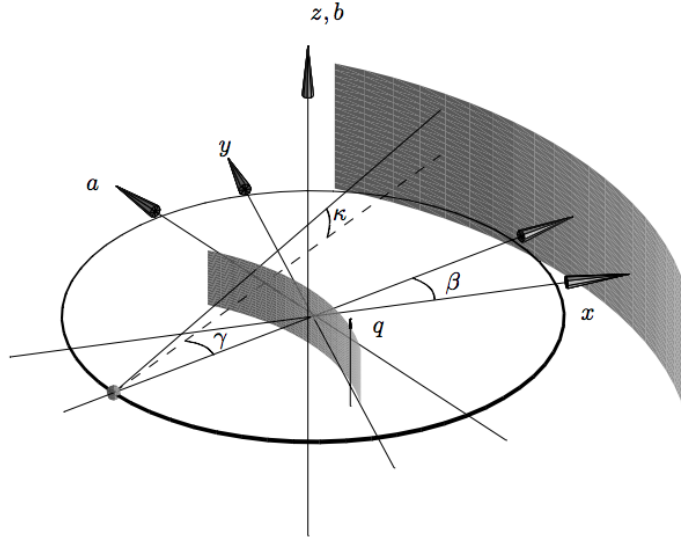


Figure 2: The cone-beam geometry [6]

3.2 Amsterdam Shroud

Motion introduced by respiration is a great factor of image quality in image-guided radiotherapy when treating a tumor which has a position that is time-variant. The ability to extract respiratory signal associated with a patient with the CBCT projections is important for lung cancer radiotherapy and substantially reduces motion blurring. As opposed to relying on an external surrogate for the respiratory signal, with the Amsterdam Shroud [4] method a signal can be extracted directly from the CBCT projections.

The procedure of the Amsterdam Shroud method is computationally simple. Firstly, the derivative of each projection is taken in the direction of the tumor movement (typically the superior-inferior direction) in order to enhance the anatomical features present in the project, then the 2D projection is converted into a 1D horizontal slice by summing over all the pixel intensities row-wise. These stack of 1D slices are placed next to each other in order of acquisition, this results in an image called the AS image. Another directional derivative (in direction of the tumor movement) is performed on the AS image in order to highlight the moving component. This can be

seen in figure 3. The respiratory signal, $I(t)$, can then be extracted from the AS image. Further, post-processing can be done to the signal in order to remove background noise.

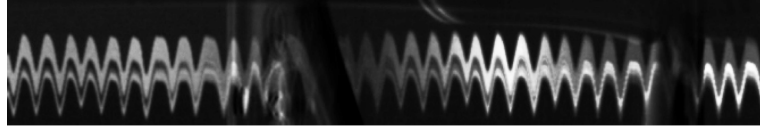


Figure 3: Extracted respiratory signal

Once the respiratory signal is extracted, it can then be broken up into ten bins phase-wise. (see figure 4). Projections belonging to a particular phase are sorted together into breathing phase bins and then CBCT images at different phases are reconstructed.

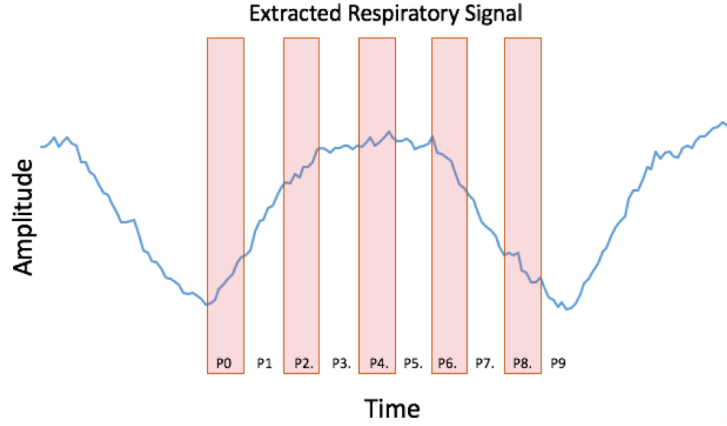


Figure 4: Respiratory signal phase binned

3.3 Data Acquisition

Phantoms

A thorax phantom was used for simulating the motion of the tumor in motion for the data acquisition. The Phantoms were the Quasar Respiratory Motion Phantom (Modus Medical Devices Inc, London, Canada) and the Dynamic Thorax Phantom (Computerized Imaging Reference Systems (CIRS), Norfolk, VA, USA), these phantom had tumor inserts with a higher density than that surrounding material which allows it to be easily seen in the X-ray scan and CT scan. The CBCT data was acquired with an Elekta Synergy Linac imaging system (XVI Systems, Elekta AB, Stockholm, Sweden) which has a CBCT x-ray source and flat-panel detector on the gantry such that they are perpendicular to the therapeutic beam. In order to convert the reconstructions to the appropriate Hounsfield units Electron Density CT phantom RMI 467 (Gammex RMI, Middleton, WI, USA) multi-density insert phantom was used.

Three scans of the Electron Density CT phantom were acquired. Two scans of the varied field-of-view (FOV),

from small to medium. the third scan repositioned the phantom such that the reconstruction had no air gap between the first slice and the last slice. Detailed parameters values are shown in table1.

Eleven scans were acquired, three static 3D CBCT of the Dynamic Thorax phantom as a control and to test that the FDK reconstruction could be performed with strong results, three 4D CBCT of the Dynamic Thorax phantom in simple motion and varying other scanning parameters, a 4D CBCT of the Dynamic Thorax phantom in irregular motion, again varying other parameters. Simple movement is where the tumor moves only in the XY plane (where X is the patient left-right and Y is superior and inferior) in the straight line. Irregular movement is where the tumor moving pattern is real patient repository data (data courtesy of University of Lübeck’s Signals @ ROB Project, Erust(2011)[7]). The tumors exhale position is not necessarily the same as its initial exhale position. These scans were taken to be comparable to the Lee at al. study[2] on image quality and tumor motion reconstruction in 4D CBCT. Detailed parameters values are shown in table2 and table 3.

Table 1: Parameters of 3D CBCT for Electron Density Phantom

Phantom	Electron Density CT	Electron Density CT	Electron Density CT
Collimation	S10	M10	M10
Filter	F0	F1	F1
Tube Voltage	120kVP	120kVP	120kVP
Tube Current	40mA	40mA	40mA
Pulse Width	40 ms	40 ms	40 ms
Number of projection	206	348	340
Total Scan Time	37.24 seconds	63.17 seconds	61.71 seconds
Full or Partial (Gantry arc)	Partial	Full	Full

Table 2: Parameters of 3D CBCT for control CIRS Phantom

Scanning Parameters	3D-Small FOV (Control)	3D-Small FOV (Control)	3D-Medium (Control)
Phantom	CIRS Dynamic	CIRS Dynamic	CIRS Dynamic
Tumor Plug	1cm	2cm	2cm
Phantom Motion	No Motion	No Motion	No Motion
Breathing Amplitude	0	0	0
Breathing Period	0	0	0
Collimation	S20	S20	M20
Filter	F1	F1	F1
Tube Voltage	120kVP	120kVP	120kVP
Tube Current	40 mA	40 mA	40 mA
Pulse Width	16 ms	16 ms	16 ms
Number of projection	373	413	641
Scan Time (Mintues)	1.13	1.25	1.99
Full or Partial (Gantry arc)	Partial	Partial	Full

Table 3: Parameters of 3D CBCT for control CIRS Phantom

FOV	Small	Small	Medium	Medium	Small	Small	Medium	Medium
Scan type	4D-Slow	4D-Fast	4D-Slow	4D-Fast	4D-Slow	4D-Fast	4D-Slow	4D-Fast
Phantom	CIRS Dynamic	CIRS Dynamic	CIRS Dynamic	CIRS Dynamic	CIRS Dynamic	CIRS Dynamic	CIRS Dynamic	CIRS Dynamic
Tumor Plug	2cm	2cm	2cm	2cm	2cm	2cm	2cm	2cm
Phantom Motion	Cos^4	Cos^4	Cos^4	Cos^4	Irregular motion	Irregular motion	Irregular motion	Irregular motion
Breathing Amplitude	$\pm 10mm(20mm)$	$\pm 10mm(20mm)$	$\pm 10mm(20mm)$	$\pm 10mm(20mm)$	$\pm 10mm(20mm)$	$\pm 10mm(20mm)$	$\pm 10mm(20mm)$	$\pm 10mm(20mm)$
Breathing Period	4 seconds	4 seconds	4 seconds	4 seconds	4 seconds	4 seconds	4 seconds	4 seconds
Collimation	S20	S20	M20	M20	S20	S20	M20	M20
Filter	F1	F1	F1	F1	F1	F1	F1	F1
Tube Voltage	120kVP	120kVP	120kVP	120kVP	120kVP	120kVP	120kVP	120kVP
Tube Current	20mA	20mA	20mA	20mA	20mA	20mA	20mA	20mA
Pulse Width	16 ms	16 ms	16 ms	16 ms	16 ms	16 ms	16 ms	16 ms
Number of projection	716	650	2385	1226	753	669	2397	1218
Scan Time (Mintues)	2.17	1.97	7.25	3.72	2.29	2.08	7.28	3.70
Gantry arc	Partial	Partial	Partial	full	Partial	Partial	full	full

3.4 Sorting XVI Output

The output from the Elekta imaging system is a series of *.HIS files, *.XML file, *.SORT.TXT file, and others. The *.HIS files are individual x-ray projection images and can be visualized in ImageJ as 16-bit Unsigned RAW files with: a width and height of 512 pixels, Offset of 100 bytes (for header information), and little-endian byte order. Metadata of each scan is found in the *.XML file, the meta-data includes Linac information, Patient name, DicomUID, Image information, individual projection (frame) information. The individual projection is of particular interest as it gives the time at which a given project was taken, the gantry angle, and more. This meta-data is later used in the reconstruction process. The *.SORT.TXT file lists the XVI binning of the respiratory cycle. The data is used to resort the projections for the individual reconstruction.

Sample of *.SORT.TXT file contains

Table 4: Sample Data from *.SORT.TXT file

Sequence	Sequence	XVI - Raw Binning number	Bin Number	
0	0	1.934909701	3	-1
1	1	2.680057526	4	-1
2	2	2.598772049	4	-1
3	3	3.076438427	5	-1
4	4	3.276117563	5	-1
5	5	3.815544367	6	-1
6	6	4.091284275	7	-1
7	7	4.42036581	7	-1

3.5 Reconstruction Toolkit

The Reconstruction Toolkit (RTK) is an open-source Toolkit cone-beam CT reconstruction built on the Insight Toolkit (ITK) and developed by the RTK consortium[8]. The RTK Package is the primary reconstruction

software used in this study. The two tools from the toolkit used are 'rtkelektasynergygeometry' which is used to convert the XVI *.XML meta-data file into an RTK geometry and the filtered back projection tool 'RTKFRK' tool is used to perform the reconstruction. 'RTKFDK' was modified to give it the ability to perform the Hounsfield conversion. The snippets of code added to the RTKFDK application can be found in the appendix.

Install code

The Reconstruction Toolkit can prove a challenge to install. Before installing RTK there are a few tools that need to be installed, namely, GIT, cmake, and gcc (or other C/C++ compiler). They can all be downloaded and installed easily using pip. The command needed is simply:

```
1 pip install git
2 pip install cmake
3 pip install gcc
```

Adding to code from Danny Lessio (member of the RTK community) I've attached a one-click install program to download ITK and RTK and install them. (see appendix)

Phase Reconstruction Code

The Phase reconstruction code reads the *.SORT.TXT and *.XML files to create 10 new directories. Using the 'Sequence' and 'bin#' columns in *.SORT.TXT copies the *.HIS to the appropriate directories. Once all the projects are sorted a new XML meta-data file needs to be created to only have the frame information for each bin. This is done by parsing the XML in python and copying the header information to 10 new files (lines 1 -38 in XML file from appendix) and the Frame tags are moved to the appropriate new XML file as per the 'Sequence' and 'bin#' columns in *.SORT.TXT. When the new XML files are filled, they are written to disk and moved to their appropriate directories.

Then using rtkelektasynergygeometry each XML is converted into RTK geometry with the command:

```
1 ITK-RTK/RTK-bin/bin/rtkelektasynergygeometry\
2 --XML \${XMLLOC}\backslash
3 -o \${RtkGeoOutLoc}
```

Then each phase in reconstructed use RTKFDK with the command:

```
1 ITK-RTK/RTK-bin/bin/rtkfdk \
2 --lowmem \
3 --geometry \${RtkGeoOutLoc} \
4 --path \${IMGLOC} \
5 --regexp '.*.his' \
6 --output \${ReconOut} \
7 --verbose \
8 --spacing \${Space},\${Space},\${Space} \
9 --dimension \${Dim},\${Dim},\${Dim}
```

"\$XML" is the location of the XML file, "\$RtkGeoOutLoc" is the location the RTK geometry output, "\$IMGLOC" is the location of the *.HIS files. "\$ Dim" and "\$Space" are constants, dimension and voxel size respectively.

4 Image Quality Metrics

The image quality of a 4D CBCT scan can be affected by a wide array of factors such as reconstruction artifacts and motion blurring, and others. Reconstruction artifacts are namely caused by having an unsatisfactory number of projections for the reconstruction. This effect can be most pronouncedly be seen in 4D scans with 650-750 projections as, after sorting, each bin is left with less than 65-75 projections. When compared to a 4D scan with 1200 projections, this effect is mitigated. In this study, in order to assess the image quality, we use the following metrics: Signal-to-noise ratio (SNR), contrast-to-noise ratio (CNR) and tumor center of mass position (TCMP).

4.1 Signal-to-Noise Ratio

the SNR is a metric to evaluate the ratio of the signal we desire to the signal of the background noise in a given region of interest (ROI). The SNR was computed by the following [3]:

$$SNR = \frac{\bar{x}_{ROI}}{\sigma_{ROI}} \quad (8)$$

Here $\bar{x}_{ROI}$ and σ_{ROI} are the mean and the standard deviation of the voxel (3D pixel) intensities in a $5 \times 5 \times 1mm^2$ ROI (See figure 5). the SNR was calculated for each of the 10 phases of the 4D CBCT image. A higher SNR value is evidence of lowering reconstruction artifacts.

4.2 contrast-to-Noise Ratio

The CNR is a metric to evaluate the visibility (contrast) of the signal we desire to the signal of the background noise in a given region of interest (ROI). The CNR was computed by the following [3]:

$$CNR = \frac{\bar{x}_{Tumor-ROI} - \bar{x}_{Background-ROI}}{\sqrt{\sigma_{Tumor-ROI}^2 + \sigma_{Background-ROI}^2}} \quad (9)$$

As is the case of the SNR analysis the ROI was the voxel intensities in a $5 \times 5 \times 1mm^2$ region (See figure 5). A higher CNR value is evidence of greater tumor visibility and is desired.

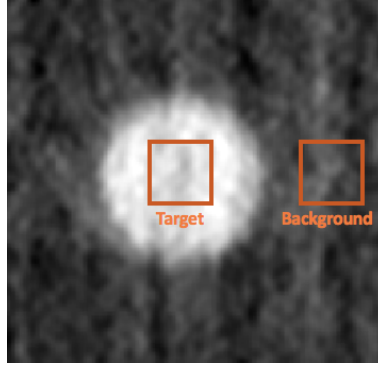


Figure 5: Highlighting the ROI used for SNR and CNR analysis

4.3 Tumor Center of Mass Position

The TCMP is a measure of the position of the tumor in a given phase, this can be compared to the expected position of the tumor in order to determine the fidelity of the tumor motion and sorting/bin processes. The TCMP is drawn in an oval to the tumor and calculates the position of the center of mass of the tumor. This analysis was only done on scan with a breathing pattern of $\cos^4$. The changing position of the tumor can be seen in figure 6.

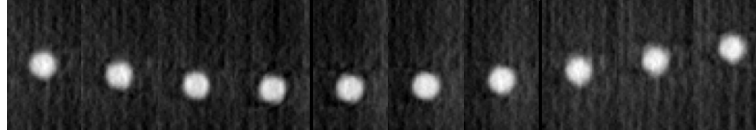


Figure 6: Changing position of the tumor from phase 1 to 10

A root-mean-square-error (RMSE) was computed by:

$$RMSE = \sqrt{\frac{1}{N} \sum_{n=1}^N (S_{Tumor}(n)^2 - S_{Reference}(n)^2)} \quad (10)$$

where N is the total number of phases in the 4D CBCT scan, $S_{Tumor}(n)$ is the position of the tumor and $S_{Reference}(n)$ is the expected position of the tumor.

5 Results

5.1 Hounsfield Conversion

The Hounsfield conversion was achieved by taking a ROI mean of each insert of the Electron Density CT phantom for both the RTK reconstruction and the XVI system reconstruction and linearly comparing the two sets of values (see table 5). The ROI of each insert can be seen in figure 7.

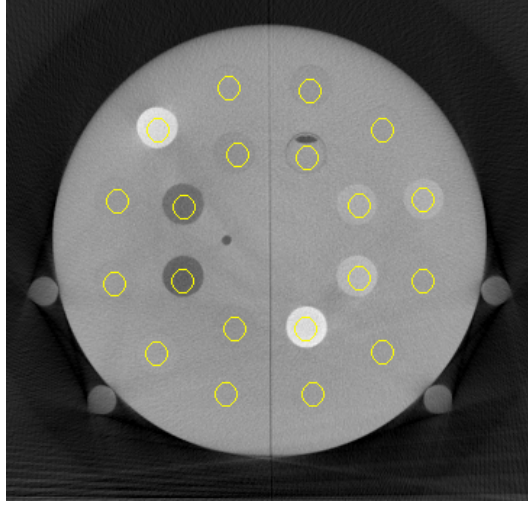


Figure 7: Electron Density Phantom and ROIs

The plot of the two sets of values and a linear fit can be shown in figure 8.

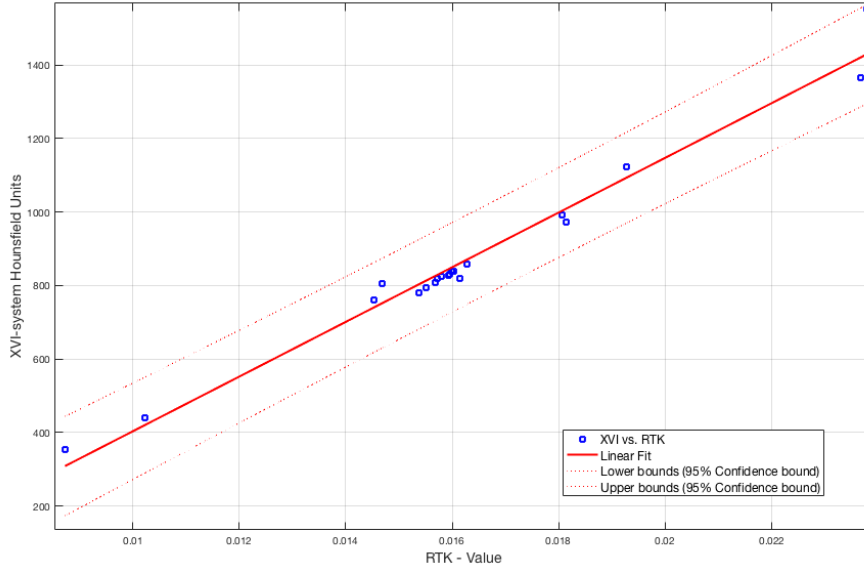


Figure 8: linear fit of the XVI Hounsfield and RTK reconstruction

The Linear fit of the XVIHU and the RTK units is found to be $f(x) = (6.865 \times 10^4) * x - 255.7$, thus the scale and shift applied to the RTK reconstruction is:

$$HU = (6.865 \times 10^4) \times \mu_{RTK_{CBCT}} - 255.7 \quad (11)$$

With a 95% confidence bounds on the scale; resulting in $[6.494 \times 10^4, 7.236 \times 10^4]$. A 95% confidence bounds on the shift resulting in: $[-315.6, -195.8]$. This fit has a sum of squared errors of prediction (SSE) of 8876, R-square value of 0.989, RMSE of 22.85.

Then this conversion is applied to the reconstruction and fitted again in order to verify that it remains a good linear fit between the XVI-system and the RTK-corrected Hounsfield units. That fit was found to be $f(x)1.014 \times [0.9549, 1.073]x - 8.292[-58.68, 42.08]$, indeed a good linear fit; which is confirmed by a SSE value of 1.025×10^4 , and R-square of 0.9873.

Table 5: HU comparison between XVI-Reference and RTK-CBCT images of the Electron Density CT phantom

Material insert	HU value (mean $\pm$ SD)		
	XVI - Reference	RTK - Uncorrected	RTK - Corrected
True Water	830 $\pm$ 18	0.01596 $\pm$ 0.00185	839 $\pm$ 37
Inner Bone	992 $\pm$ 21	0.01807 $\pm$ 0.00175	972 $\pm$ 34
CB2 - 30%	1124 $\pm$ 25	0.01927 $\pm$ 0.00189	1065 $\pm$ 37
Cortical Bone	1552 $\pm$ 25	0.02378 $\pm$ 0.0019	1372 $\pm$ 39
LN-300 Lung	355 $\pm$ 16	0.00873 $\pm$ 0.00168	348 $\pm$ 36
LN-450 Lung	442 $\pm$ 20	0.01023 $\pm$ 0.00186	439 $\pm$ 36
Adipose	761 $\pm$ 21	0.01453 $\pm$ 0.00173	746 $\pm$ 34
Breast	781 $\pm$ 19	0.01537 $\pm$ 0.00165	802 $\pm$ 33
Solid Water	827 $\pm$ 21	0.01594 $\pm$ 0.00169	840 $\pm$ 34
B-200 Bone	972 $\pm$ 18	0.01813 $\pm$ 0.00157	996 $\pm$ 36
Liver	858 $\pm$ 18	0.01627 $\pm$ 0.00183	856 $\pm$ 36
Brain	809 $\pm$ 18	0.01569 $\pm$ 0.0016	819 $\pm$ 33
CB2 - 50%	1365 $\pm$ 17	0.02367 $\pm$ 0.00175	1349 $\pm$ 40

5.2 SNR

Figures 9, 10, and 11 shows the the SNR value for eight scans from table 3. The scans for Medium FOV and slow speed are barred from further consideration as their acquisition time (7.50 minutes) and the number of projections (2500) is too great. A seven-minute scan in a clinical setting would not be reasonable. The greatest mean values for both regular and irregular motion in the SNR of the lung (background signal), SNR of the Tumor, and the CNR are achieved from the medium FOV Fast scan. (see table 6)

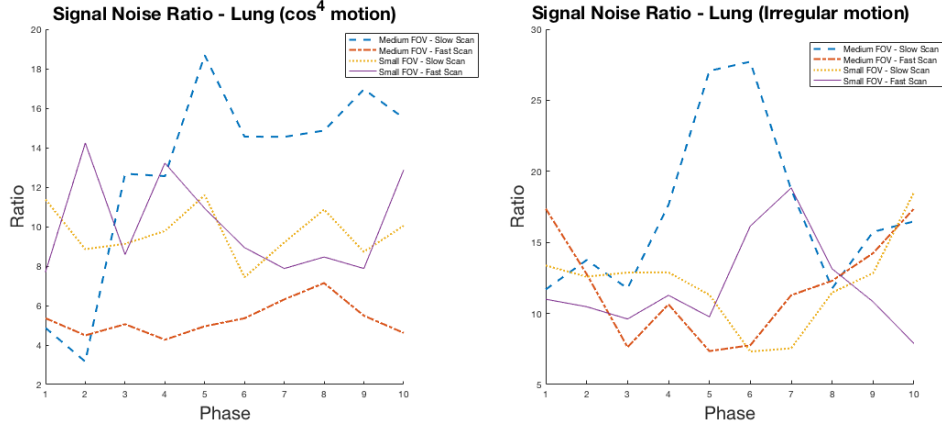


Figure 9: SNR of lung for regular and irregular motion in each phase

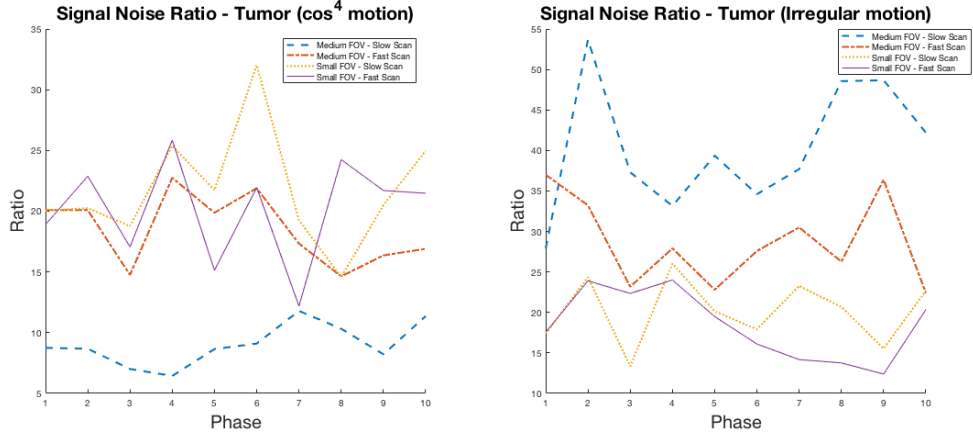


Figure 10: SNR of tumor for regular and irregular motion in each phase

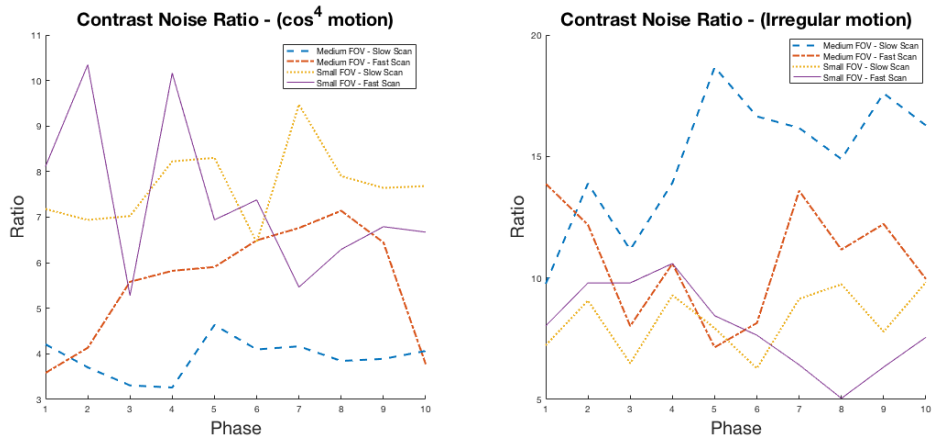


Figure 11: CNR for regular and irregular motion in each phase

Table 6: SNR and CNR Mean values for regular and irregular motion

Breathing Pattern	FOV - Speed	SNR Lung Mean	SNR Tumor Mean	CNR Mean
Cos^4	Medium - Slow	11.23	5.55	3.92
	Medium - Fast	9.70	21.75	7.68
	Small - Slow	5.30	18.47	5.56
	Small - Fast	10.07	20.14	7.34
Irregular	Medium - Slow	17.23	40.31	14.90
	Medium - Fast	11.87	28.72	10.69
	Small - Slow	12.07	20.13	8.28
	Small - Fast	11.90	18.41	7.97

5.3 Tumor Center of Mass Position

The 4D CBCT reconstruction of all images were of dimensions of $270 \times 270 \times 270$ where each voxel is $1mm^3$. This leads to a measurement uncertainty of $\pm 1mm$ on all measurements. The Position of the tumor for a Medium FOV fast scan of regular motion was fitted with the following function:

$$f(x) = a * \cos(b * x - c)^4 + d \quad (12)$$

Figure 12 shows that the center of mass position is within the 95% confidence bounds of the fit. The fit has an SSE of 4.972, R-square of 0.9828, and RMSE of 0.9972.

Here we can say with strong certainty that the tumor indeed was positioned where we would expect it.

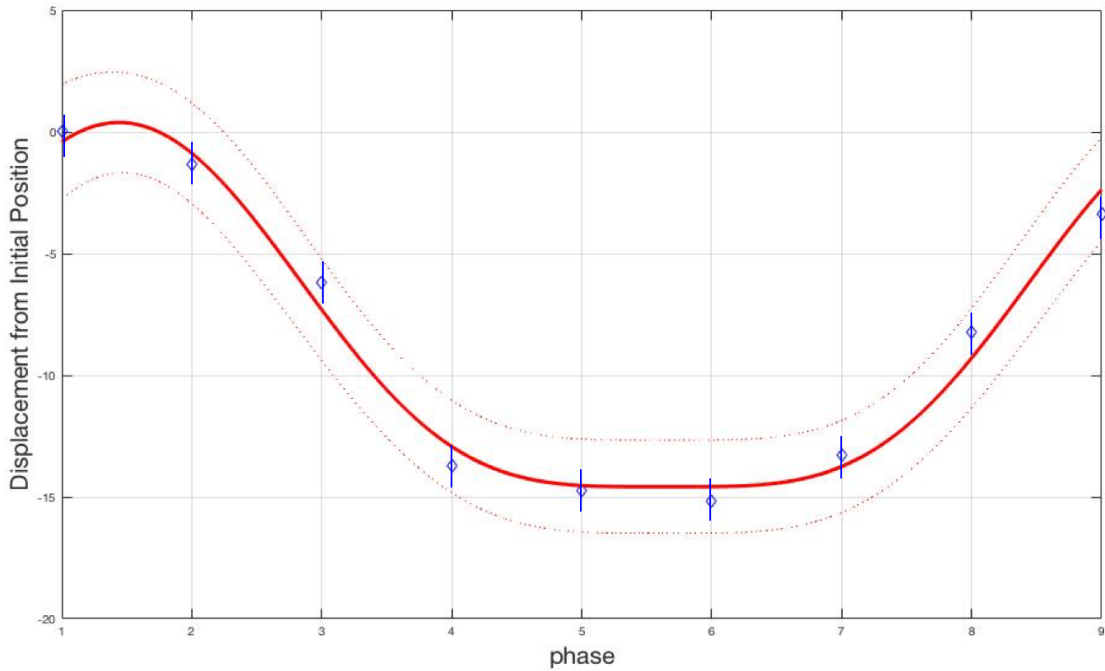


Figure 12: Position of tumor

6 Discussion

In this study, we conducted an initial assessment of the the image quality of the Elekta XVI 4D CBCT system. This began by building a framework to perform 4D CBCT reconstructions off-line, for this task the RTK tool kit was employed. It was installed on Carleton Physics' Unix cluster which allowed for greater computation power and remote access to the tool kit. RTK's FDK reconstruction applications as packaged needs to be modified in order to allow for Hounsfield conversion natively, as such a Hounsfield conversion tool was built it. A 3D median filter from the ITK tool kit was also built into the applications. The RTK tool kit outputs files in the ITK standard *.MHA, which can be opened with Fiji [11] for analysis. the Elekta XVI system outputs its reconstitution in a *.SCAN file which can be converted into the ITK standard with VV [13], a 4D slicer tool. Once converted the XVI reconstitution can be compared to the RTK reconstitution.

The medium FOV and slow speed scans were acquired for the sake for comparison and as suggested in Impact of scanning parameters and breathing patterns on image quality and accuracy of tumor motion reconstruction in 4D CBCT [2]. However, these scans were not considered as their acquisition time (7.50 minutes) and the number of projections (2500) is too great. It would not be reasonable in a clinical setting to acquire a scan with these acquisition times and number of projections, the time cost and radiation delivered to the patient would render this scan impractical.

7 Conclusions

In the course of this study we successfully acquired the first 4D CBCT images at the Ottawa Hospital Cancer Center. To allow for off-line study and analysis of these images we built a 4D CBCT reconstruction process using RTK and compared it to the clinical XVI system.

Although we did not conduct our own phase sorting and instead relied on the the sorting to be performed by the XVI system, we were able to conclude that the position of the tumor, after sorting/binning, is in the expected position (95% confidence bounds).

Through our analysis we are able to conclude that the best image quality within a reasonable acquisition time and number of projections was with the medium FOV "Fast" acquisition (1218 Projections, 3.7 minutes). In this time frame it provides a reasonably good image for the purposes of patient positioning.

8 Appendix

Sample of *.XML file contains

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ProjectionSet>
3    <!--Do not edit this file.-->
4    <Station>
5      <StationName>XVI_UNIT_8</StationName>
6      <text>YOOOOOOOOO</text>
7      <LinacID>1927</LinacID>
8    </Station>
9    <Patient>
10     <FirstName>TEST1</FirstName>
11     <MiddleName></MiddleName>
12     <LastName>_PENTAGUIDE</LastName>
13     <ID>RPENTA01</ID>
14   </Patient>
15   <Treatment>
16     <ID>1: Align to centre marks</ID>
17     <Description>AgilPGaligncen: AgilPGaligncen</Description>
18     <DicomUID>2.16.840.1.114337.141568264680008.16230.1393268348.0</DicomUID>
19   </Treatment>
20   <Field>
21     <Id>***KV-IMAGES***</Id>
22     <Description>***KV-IMAGES TREATMENT:1378614</Description>
23   </Field>
24   <Image>
25     <kV>120</kV>
26     <mA>20</mA>
27     <ms>16</ms>
28     <AcquisitionPresetName>5.0Symmetry</AcquisitionPresetName>
29     <Width>512</Width>
30     <Height>512</Height>
31     <Depth>16</Depth>
32     <DicomUID>1.3.46.423632.1354522017101323384362.77</DicomUID>
33     <CTDIvol>12.2</CTDIvol>
34     <CTDIPhantomType>Body Phantom (Length 40cm)</CTDIPhantomType>
35     <AbsoluteTableLatPosIEC1217_MM>-83.6</AbsoluteTableLatPosIEC1217_MM>
36     <AbsoluteTableLongPosIEC1217_MM>210.3</AbsoluteTableLongPosIEC1217_MM>
37     <AbsoluteTableVertPosIEC1217_MM>-122.1</AbsoluteTableVertPosIEC1217_MM>
38   </Image>
39   <Frames>
40     <Frame>
41       <Seq>1</Seq>
42       <DeltaMs>0</DeltaMs>
43       <HasPixelFactor>False</HasPixelFactor>
44       <PixelFactor>0</PixelFactor>
45       <GantryAngle>-180.014282227</GantryAngle>
46       <Exposed>True</Exposed>
```

```
47      <MVOnt>False</MVOnt>
48      <UCentre>-3.792999983</UCentre>
49      <VCentre>2.259000063</VCentre>
50      <Inactive>False</Inactive>
51    </Frame>
52    ...
```

Auto Install RTK Reconstruction toolkit Package

```
1  # Author: Waqar Muhammad – Carleton Physics 2018
2  # Written in: Bash
3  # Project Repository: https://github.com/WaqarSM/4DCBCT-Code-Repository
4  #
5  # Disclaimer: Part of this code was written by Danny Lessio for the RTK consortium.
6
7
8  if [ ! -d "./RTK-ITK-VTK" ]; then
9      #making a directory to host RTK ITK and VTK
10     mkdir ITK-RTK
11     cd ITK-RTK
12     echo ITK-RTK directory made!
13 fi
14
15 if [ ! -d "./VTK" ]; then
16     git clone https://github.com/Kitware/VTK.git vtk
17     git checkout v7.0.0
18     echo VTK directory made, and gitcloned (v7.0.0 checked out)
19 fi
20
21 if [ ! -d ".VTK-bin" ]; then
22     # Compiling VTK
23     mkdir VTK-bin
24     cd VTK-binary
25     cmake -DModule_vtkGUISupportQt=ON -DModule_vtkGUISupportQtOpenGL=ON -D -
26         DModule_vtkRenderingQt=ON -DModule_vtkViewsQt=ON -DVTK_RENDERING_BACKEND=OpenGL -
27         DVTK_QT_VERSION=5 ../VTK
28     make -j4
29     path_of_VTK=$(pwd) # Getting the full path of VTK-bin
30     cd ..
31 fi
32
33 if [ ! -d ".ITK" ]; then
34     git clone git://itk.org/ITK.git
35     echo ITK directory made, and gitcloned!
36 fi
37
38 if [ ! -d ".ITK-bin" ]; then
39     # Compiling ITK
40     mkdir ITK-bin
41     cd ITK-bin
42     cmake -DModule_ITKReview=ON -DITK_USE_FFTWD=ON -DVTK_DIR=$path_of_VTK -DITK_USE_FFTWF=ON -
43         DBUILD_DOCUMENTATION=OFF -DBUILD_EXAMPLES=OFF -DBUILD_TESTING=OFF -DCMAKE_CXX_FLAGS=
44         fPIC -DCMAKE_C_FLAGS=-fPIC ../ITK
45     make -j4
46     path_of_ITK=$(pwd) # Getting the full path of ITK-bin
47     cd ..
48     echo ITK-bin directory made, and made!
```

```

45  fi
46
47  if [ ! -d "./RTK" ]; then
48      git clone git://github.com/SimonRit/RTK.git
49      echo RTK directory made, and gitcloned!
50  fi
51
52  if [ ! -d "./RTK-bin" ]; then
53      # Compiling RTK
54      mkdir RTK-bin
55      cd RTK-bin
56
57      cmake -DITK_DIR=$path_of_ITK -DCMAKE_BUILD_TYPE=Release -DBUILD_EXAMPLES=ON -
          DCMAKE_CXX_FLAGS=-fPIC -DCMAKE_C_FLAGS=-fPIC ../RTK
58      make -j4
59      echo ITK-bin dic made, and made!
60      cd bin
61      ./HelloWorld
62      cd ..
63      cd ..
64  fi

```

Sorting and Reading new XML Files code

```
1  # Author: Waqar Muhammad – Carleton Physics 2018
2  # Project Repository: https://github.com/WaqarSM/4DCBCT-Code-Repository
3  # Written in: Python (Tested in python 2.7)
4  #
5  # This code will read in the *.HIS file , XML file , and the *.SORT.txt file
6  # It outputs sorted/binned *.HIS and new XML files
7
8  #importing libraries
9  import csv
10 import xml.etree.ElementTree as ET
11 import numpy as np
12 import array
13 import shutil
14 import os
15
16 #—————File Locations—————
17 #This snippet sets the location of the original *.HIS files , *.SORT.txt file , *.XML file and
    output location of binned data
18 # Change the hisFileLoc to the location of the *.HIS file Projections
19 hisFileLoc= '/Users/mwaqar/4DCBCT_Project/img_1.3.46.423632.1354522017101323933437.72/' #
    Location of the directories that contains the original *.HIS files – DOES need to be
    changed
20 sortFileLoc = hisFileLoc+'Reconstruction
    /1.3.46.423632.1354522017101323933437.72.10.13.2017191452671.SCAN.Sort.txt' #Location of
    the *Sort.txt file – does not need to be changed
21 xmlFileLoc= hisFileLoc+'_Frames.xml' #Location of XML file – does not need to be changed
22 sortedFileLoc = hisFileLoc+'sorted' #Location of binned data output – does not need to be
    changed.
23
24 #—————Parsing Sort file—————
25 #This snippet Parses the *.SORT.txt file
26 with open(sortFileLoc , 'rb') as csvfile: #calls open to read (in binary) to parse the *.SORT.
    txt file
27     csvreader = csv.reader(csvfile , delimiter='\t' , quotechar='|') #setting parsing
        parameters, tabs as the delimitation and 'enter' as a new row
28     sortList=[] #decalering array to hold Bin numbers
29     for i in csvreader: #For loop to iterate .SORT.txt file and append Bin number to array
        sortList
30         sortList+=i[3]
31
32     print '————'
33     sortListInt = [int(x) for x in sortList]
34     print sortListInt
35     print 'Length if sort list =', len(sortListInt)
36
37 #—————Parsing Elekta XML—————
38 #For contest please seethe Elekta *.XML file
39 tree = ET.parse(xmlFileLoc) #Parsing Elekta XML using the xml.etree.ElementTree library as
```

```

    tree
40 root = tree.getroot() #Setting first degree tags as root
41 print root.tag #Printing all the tags in root
42 print root.attrib #printing attributes of tags in root
43 print "———"
44 for child in root:
45     print child.tag, child.attrib # print all tags under ProjectionSet
46 print "———"
47 print root[4][7].tag
48 print root[4][7].text #perfix of DicomUID Name
49
50 suffixName = root[4][7].text #suffix of DicomUID Name
51 frameXMLItem=root.findall('Frames/Frame') #parsing all Frame elements in Frames
52
53 ET.dump(frameXMLItem[0]) #Printing Frame number
54 print "———"
55
56 for item in frameXMLItem:
57     print item.find('Seq')
58     print item.find('Seq').text
59
60 # —————Making new directories for sorted files and Copying *.HIS files to new
    directories —————
61 print "—Naming—"
62 fileNameInt = range(1,len(sortListInt)+1) #making a seq. array for the number of files
63 print fileNameInt
64 print len(fileNameInt)
65
66 fileNameZintl = [0] * len(fileNameInt) #Making empty array size equal to # of files
67 for i in fileNameInt:
68     fileNameZintl[i-1] = str(fileNameInt[i-1]).zfill(5)+'.'+suffixName+'.his' # Adding the
        0000 prefix to the same
69 #print fileNameZintl
70 #print len(fileNameZintl)
71
72 print "—Moving—"
73 if not os.path.exists(sortedFileLoc): #checking if sort directories exist if not makes
    directories
74     os.makedirs(sortedFileLoc)
75 else:
76     print "dir already exists at "+sortedFileLoc #else outputs that they exist
77
78 pathIndex=range(0,10)
79 for i in pathIndex:
80     if not os.path.exists(sortedFileLoc+"/"+str(i)): #checking if sort directories exist if
        not makes 10 new directories
81         os.makedirs(sortedFileLoc+"/"+str(i))
82     else:
83         print "dir already exists at /sorted/"+str(i)

```

```

84
85 HisFileFromLoc = [0] * len(sortListInt) #array to hold *.HIS file names
86 for i in range(0,len(sortListInt)): #Moving (well making copies of) to bins
87     print i
88     movedLoc=sortedFileLoc+"/"+str(sortListInt[i])+"/"+fileNameZintl[i] #variable = location
89         of binned *.HIS + *.HIS file name
90     print movedLoc
91     HisFileFromLoc[i]=hisFileLoc+fileNameZintl[i] #Location on original *.HIS file + name
92     print HisFileFromLoc[i]
93     print "-"
94     shutil.copyfile(HisFileFromLoc[i], movedLoc) #Making a Copying to *.HIS file to bin
95         directory
96
97 #-----Making sorted XML file -----
98 #This spinnet creates XML files for the binned data
99 stationInfo=root.find('Station') #object holding all Station element info
100 patientInfo=root.find('Patient') #object holding all Patient element info
101 treatmentInfo=root.find('Treatment') #object holding all Treatment element info
102 fieldInfo=root.find('Field') #object holding all Field element info
103 imageInfo=root.find('Image') #object holding all Image element info
104 FrameInfoList=root.findall('Frames/Frame') #object holding all Frame tag info (n.b. Frames is
105     not to be confused with Frame; there are many frame elements and they are sub-elements of
106     Frames )
107 ET.dump(FrameInfoList[0])
108
109 # ProjectionSetSortList = ET.Element('ProjectionSet')
110 # ProjectionSetSortList.append(stationInfo)
111 # ProjectionSetSortList.append(patientInfo)
112 # ProjectionSetSortList.append(treatmentInfo)
113 # ProjectionSetSortList.append(fieldInfo)
114 # ProjectionSetSortList.append(imageInfo)
115
116 ProjectionSetSortList=[0]*10
117 for i in range(0,10):
118     ProjectionSetSortList[i] ='ProjectionSetSort'+str(i)
119     print ProjectionSetSortList[i]
120     ProjectionSetSortList[i] = ET.Element('ProjectionSet') #making a XML main tag called
121         Projection set
122     ProjectionSetSortList[i].append(stationInfo) #Appending station element Info
123     ProjectionSetSortList[i].append(patientInfo) #Appending patient element Info
124     ProjectionSetSortList[i].append(treatmentInfo) #Appending treatment element Info
125     ProjectionSetSortList[i].append(fieldInfo) #Appending field element Info
126     ProjectionSetSortList[i].append(imageInfo) #Appending image telementag Info
127     framesInfo=ET.SubElement(ProjectionSetSortList[i], 'Frames') #Appending element called
128         Frames to hold binned Frame tags (n.b. Frames is not to be confused with Frame; there
129         are many frame elements and they are sub-elements of Frames )
130     # ProjectionSetSortList[i].append(framesInfo)
131     myXMLdataPYloop =ET.tostring(ProjectionSetSortList[i]) #prepping data of XML file to be
132         saved

```



```

125     myXMLfile = open(sortedFileLoc+"/"+str(i)+"/"+"_Frames.xml", "w") #opening new XML files
        at its sorted location
126     myXMLfile.write(myXMLdataPYloop) #writing XML file to new location
127
128 for i in range(0,len(sortListInt)):
129     print i
130     ProjectionSetSortList[sortListInt[i]].find('Frames').append(FrameInfoList[i]) #Finding
        binned Frame elements to Frames
131     ProjectionSetSortList[sortListInt[i]].append(FrameInfoList[i]) #Appending binned Frame
        file to Frames
132     myXMLdataPYloop =ET.tostring(ProjectionSetSortList[sortListInt[i]]) #prepping data of XML
        file to be saved
133     #myXMLfile = open(sortedFileLoc+"/"+str(sortListInt[i])+"/"++"_Frames"+str(sortListInt[i])
        + ".xml", "w")
134     myXMLfile = open(sortedFileLoc+"/"+str(sortListInt[i])+"/"++"_Frames.xml", "w")
135     myXMLfile.write(myXMLdataPYloop) #writing data to XML file

```

Reconstruction of each binned Phase using RTK

```

1  # Author: Waqar Muhammad – Carleton Physics 2018
2  # Project Repository: https://github.com/WaqarSM/4DCBCT-Code-Repository
3  # Written in: Bash
4  # This script will reconstruction every phase after its been using 4DCBCT-Resort-XMLgen.py
5  # Inputs:
6  #   RTKbinLOC – is the location of the RTK Compiled code (ie. it's binary)
7  #   FileName – The name of the scan directory
8  #   FileNum – The short version name of the scan
9  #
10 # Output:
11 #   10 reconstruction (one for each phase) location is set by RtkGeoOutLoc
12
13 startTime=$(date +%s) #begins Main Timer
14 # Note: Path location can not have any spaces.
15 FileName=img_1.3.46.423632.1354522017101323295146.75 #DICOM_ID of directory
16 FileNum=75 #Reconstruction identifier
17 Dim=270 #Dimension in millimeters
18 Space=1 #Spacing of Voxel
19 SpaceZ=1 #Z spacing of Voxel
20
21 for (( i=0; i<=9; i++)) #Main Loop – Runs reconstruction on each binned phase
22 do
23     startTimeloop=$(date +%s) #Timer for Loop
24     echo "Phase being reconstructed $i"
25     Phase=$i
26     RTKbinLOC=/data/data068/mwaqar/ITK-RTK/RTK-bin/bin/ #Location of RTK-bin
27     IMGLOC=/data/data068/mwaqar/Sorted_4DCBCT_Data/$FileName/sorted/$Phase #Location of Binned
28         *.HIS files
29     XMLLOC=/data/data068/mwaqar/Sorted_4DCBCT_Data/$FileName/sorted/$Phase/_Frames.xml #
30         Location of Binned _Frame.XML
31     RtkGeoOutLoc=/data/data068/mwaqar/Sorted_4DCBCT_Data/$FileName/sorted/$Phase/
32         elektaGeometry #Location of RTK geometry file
33     ReconName=$FileNum'_p'$Phase'_d'$Dim'_s'$Space'_sZ'$SpaceZ #Name of reconstructed file
34         with Phase, dimension ,and spacing information.
35     ReconOut=/data/data068/mwaqar/Sorted_4DCBCT_Data/$FileName/sorted/good/$ReconName'spr_24.
36         mha' #Location of reconstruction output
37
38     echo 'RTK Bin location: ' $RTKbinLOC
39     echo 'Projection source: ' $IMGLOC
40     echo 'XML File location : ' $XMLLOC
41     echo 'RTK Geo output location: ' $RtkGeoOutLoc
42     echo 'Recon output location: ' $ReconOut
43
44     echo ' _____Now Running RTK Elekta Geometry_____ '
45     /data/data068/mwaqar/ITK-RTK/RTK-bin/bin/rtkelektasynergygeometry \ #Calling geometry
46         application. written by the RTK consortium
47     —xml $XMLLOC \ #location of binned XML file
48     —verbose \ #enabling verbose
49     —o $RtkGeoOutLoc #Location of RTK geometry output

```

```

43 echo ' _____Now Running RTK FDK_____',
44   /data/data068/mwaqar/ITK-RTK/RTK-bin/bin/rtkfdk \ #Calling reconstruction application.
      written by the RTK consortium and appended to by Waqar Muhammad 2018 (Find updated
      file in project repository)
45   --lowmem \ #enabling on low memory mode
46   --geometry $RtkGeoOutLoc \ #inputting location of RTK geometry output
47   --path $IMGLOC \ #Inputting location of *.HIS file
48   --regexp '.*.his' \
49   --output $ReconOut \ #Location of reconstruction output
50   --verbose \ #enabling verbose
51   --spr=0.24 \ #scatter to primary
52   --spacing $Space,$Space,$SpaceZ \ #setting spacing of X,Y and Z
53   --dimension $Dim,$Dim,$Dim # Setting dimension of reconstruction
54   endTime=$(date +%s) #Loop Timer
55   TotalTime=$(( $endTime - $startTime )) #Loop Timer
56   echo Took $TotalTime
57 done
58 endTime=$(date +%s) #Main Timer
59 TotalTime=$(( $endTime - $startTime )) #Main Timer
60 echo Total Time: $TotalTime

```

References

- [1] Martin Beeres, Julian L Wichmann, Jijo Paul, Emmanuel Mbalisike, Mohamed Elsabaie, Thomas J Vogl, Nour-Eldin A Nour-Eldin, CT chest and gantry rotation time: does the rotation time influence image quality? (2015), *Acta Radiologica* Vol 56, Issue 8, pp. 950 - 954, doi:10.1177/0284185114544242
- [2] Lee, S., Yan, G., Lu, B., Kahler, D., Li, J. G. and Sanjiv, S. S. (2015), Impact of scanning parameters and breathing patterns on image quality and accuracy of tumor motion reconstruction in 4D CBCT: a phantom study. *Jnl Applied Clin Med Phys*, 16: 195–212. doi: 10.1120/jacmp.v16i6.5620
- [3] Thengumpallil, S., Smith, K., Monnin, P., Bourhis, J., Bochud, F. and Moeckli, R. (2016), Difference in performance between 3D and 4D CBCT for lung imaging: a dose and image quality analysis. *Jnl Applied Clin Med Phys*, 17: 97–106. doi: 10.1120/jacmp.v17i6.6459
- [4] Ahmad, M., Cervino, L., Jia, X., Jiang, S.B., Mou, X., Pan, T., Wang, X., Yan, H., & Yin, W. (2013). Extracting respiratory signals from thoracic cone-beam CT projections. *Physics in medicine and biology*, 58 5, 1447-64. doi:10.1088/0031-9155/58/5/1447
- [5] L. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone-beam algorithm," *J. Opt. Soc. Am. A* 1, 612-619 (1984)
- [6] Turbell H (2001), *Cone-Beam Reconstruction Using Filtered Backprojection*, Linkoping Studies in Science and Technology Dissertation No. 672
- [7] Floris Ernst, *Compensating for Quasi-periodic Motion in Robotic Radiosurgery*, Springer, 2011, DOI 10.1007/978-1-4614-1912-9
- [8] Rit, S.; Vila Oliva, M.; Brousmiche, S.; Labarbe, R.; Sarrut, D.; Sharp, G. C. (2014), The Reconstruction Toolkit (RTK), an open-source cone-beam CT reconstruction toolkit based on the Insight Toolkit (ITK), doi:10.1088/1742-6596/489/1/012079
- [9] Lechuga, L., & Weidlich, G. A. (2016). Cone Beam CT vs. Fan Beam CT: A Comparison of Image Quality and Dose Delivered Between Two Differing CT Imaging Modalities. *Cureus*, 8(9), e778. doi:10.7759/cureus.778
- [10] T.S. Yoo, M. J. Ackerman, W. E. Lorensen, W. Schroeder, V. Chalana, S. Aylward, D. Metaxas, R. Whitaker. Engineering and Algorithm Design for an Image Processing API: A Technical Report on ITK - The Insight Toolkit. In *Proc. of Medicine Meets Virtual Reality*, J. Westwood, ed., IOS Press Amsterdam pp 586-592 (2002).
- [11] Schindelin, J.; Arganda-Carreras, I. & Frise, E. et al. (2012), "Fiji: an open-source platform for biological-image analysis", *Nature methods* 9(7): 676-682, PMID 22743772, doi:10.1038/nmeth.2019
- [12] Rueden, C. T.; Schindelin, J. & Hiner, M. C. et al. (2017), "ImageJ2: ImageJ for the next generation of scientific image data", *BMC Bioinformatics* 18:529, doi:10.1186/s12859-017-1934-z.
- [13] Seroul, P. and Sarrut, D. VV : Viewer for the evaluation of 4D image registration. 2008. presented at the MICCAI workshop Systems and Architectures for Computer Assisted Interventions

- [14] Fedorov A., Beichel R., Kalpathy-Cramer J., Finet J., Fillion-Robin J-C., Pujol S., Bauer C., Jennings D., Fennessy F.M., Sonka M., Buatti J., Aylward S.R., Miller J.V., Pieper S., Kikinis R. 3D Slicer as an Image Computing Platform for the Quantitative Imaging Network. *Magn Reson Imaging*. 2012 Nov;30(9):1323-41. PMID: 22770690. PMCID: PMC3466397.