

Fluoro Simulator User Manual

Ju Young Ahn, Seung Wook Lee
Last Revision: 05/05/16

1. Mobile C-arm Modeling

1.1. Source Transform

The purpose of source transform is to define source position in world coordinate. The transform is currently composed of two components: orbital rotation and angular rotation. Therefore, the transformation could be defined as:

$$T_S = R_a(\alpha)R_o(\omega)$$

where R_o represents orbital rotation, and R_a represents angular rotation. Initially, source position lies on an -y-axis, R_o rotates the source about the x-axis for an angle ω , and R_a rotates the position about an axis z' , which is a rotated z-axis, for an angle α . Figure 1 describes a view of original position of the source in the system coordinate. Axis z' is defined as a line passing origin and a point $(x,y,z) = (0, -\sin \omega, \cos \omega)$, where ω is an orbital position of a C-arm.

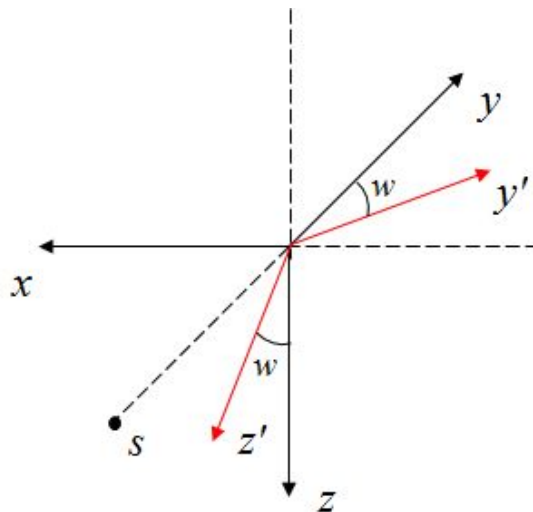


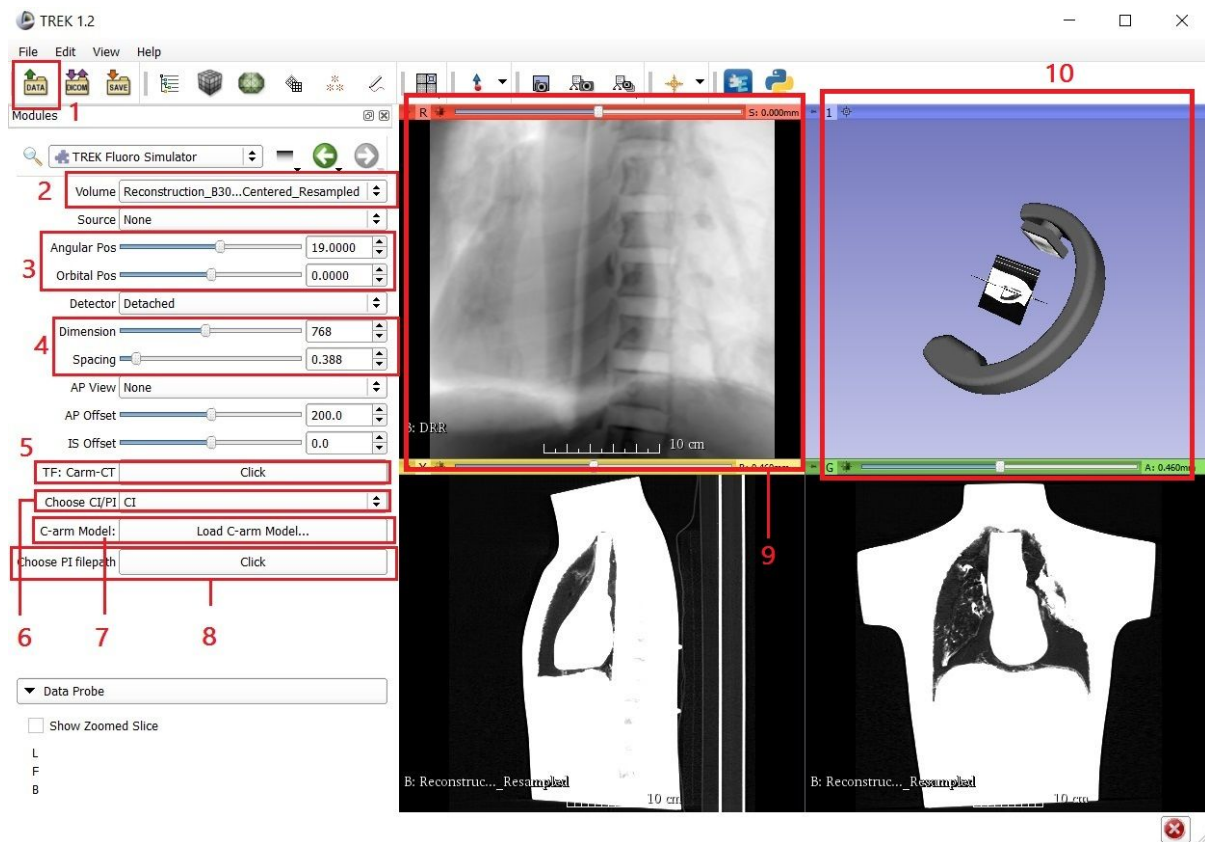
Figure 1. Schematic for coordinate and rotation of FluoroSim System

Initial source position is defined to be $(0, -612, 0)$ where the initial detector position is $(0, 488, 0)$. These positions reflected geometric properties of the C-arm of which source and detector position is approximately 612mm and 488mm apart from the center of rotation.

1.2. Detector Transformation

Detector transformation is similar to the source transformation. On an X-ray C-arm, detector and source has a fixed source-to-detector distance (SDD) and positioned at the opposite side. Therefore, the detector transformation rotates could be described as a source transformation with additional $\pi/2$ rotation in either angular or orbital direction. On our system, additional $\pi/2$ is added in angular position to define the source position.

2. UI design



<Figure 2. Screenshot of FluoroSim System>

What's below are description on functions of each red box.

1. The button allows loading of the volume, transformation, or other data files.
2. "Volume" toggle allows user to select loaded (preoperative CT) volume to display DRR
3. Slider to adjust orbital and angular positions of the C-arm. Can move manually in computer interface and reads physical position of the C-arm in physical interface.
4. Dimension of detector. Default dimension and spacing are 768 pixels and 0.388 mm per pixel. This ratio needs to be maintained to accurately display an image of the same size with an actual radiograph.
5. Load registration transform. Clicking the button displays a pop-up window to select registration transform file. Currently only support a .txt format (see Section 3.)
6. Allow user to switch between computer interface (CI) and physical interface (PI). In computer interface, a user can control a C-arm position in a digital space. In physical interface, user can control the position by physically moving the C-arm.
 - a. When PI is selected, a dialogue box will pop up and the user can select an input .txt file that contains real time angular and orbital positions of C-arm.
 - b. When PI is selected, a dialogue box will pop up and the user can select an input .txt file that contains real time angular and orbital positions of C-arm.
 - c. The module currently reads angular and orbital positions from a text file every second (Thus, the DRR is generated one frame per second)
7. Button to load C-arm model. On default, a model of Cios-alpha is automatically loaded (.STL format)


- a. Make sure to check for a correct path for the C-arm STL file which is currently hard coded.
8. This button allows user to choose output file of an encoder. Without signifying this file path, the module would not be able to read encoder position and physical position of the C-arm.
9. Window to display DRR.
10. 3D space for visualization of the C-arm and the patient. Visualizes current position of the C-arm relative to the patient. Without C-arm model properly loaded, (in part 7) the model might not show on the 3D space.

2.1 Producing and generating DRR

- DRR is generated when source position defined by sliders **changes**.
- To see DRR, click a pin-shaped button at the top left corner of red/yellow/green window. Then, select 'axial' and 'DRR' to see DRR.



<Figure 3. How to select to see a particular volume in 3D Slicer>

- If the DRR is not centered, click focus button () on the top left corner.

3. Registration and Geometric Calibration

The purpose of registration is to define relative position between the C-arm and patient. In our experiment, 3D-2D image registration is performed in a separate platform and the resulting transformation matrix is loaded to our module by selecting a file (refer to 2. UI Design 5). Format of the text file is the following:

```
|0.99949, -0.016472, 0.027371, 2.1868
0.014793, 0.99806, 0.060446, 80.667
-0.028314, -0.06001, 0.9978, -4.1029
0, 0, 0, 1
```

: Please note that the format of transformation needs to be accurate.

The matrix needs to be 4 by 4 matrix.

Due to non-isocentricity of a C-arm in its orbital rotation, C-arm model proposed in part 1 does not accurately predict source and detector position for orbital rotation. Therefore, geometric calibration (GeoCalc) data is used to more accurately simulate the geometry of mobile C-arm for purely angular and orbital positions. Currently, when orbital or angular positions are within +/- 1 degree from origin, the system loads the GeoCalc data from a specified directory as a look-up table. The GeoCalc data is defined for full 360 degrees in angular positions and -40 to +20 degrees in orbital positions.

The GeoCalc data is originally provided as a *.mat* file, but is converted to *.pickle* format using module *pickle* in Python. Current environment of TREK uses Python 2.7.10, and it was difficult to directly read *.mat* file in the environment. So the *.mat* file is converted to *.pickle* file with protocol 2.

Provided below is a sample code for conversion is following (in Python 3.5.1):

```
import pickle
import scipy.io as scio
import numpy as np
```

```
GeoCalcDict = scio.loadmat('Directory\\GeoCalcData.mat')
    with open('Directory\\GeoCalcData.pickle', 'wb') as handle:
        pickle.dump(GeoCalcDict , handle , 2)
```