ITK-Based Implementation of Two-Projection 2D/3D Registration Method with an Application in Patient Setup for External Beam Radiotherapy

Release 1.00

Jian Wu[1,2]

December 21, 2010

[1]University of Florida, Gainesville, Florida
[2]Virginia Commonwealth University, Richmond, Virginia

Abstract

This document describes an ITK-based implementation of intensity-based 2D/3D rigid image registration for patient setup assessment in external beam radiotherapy. The registration framework was designed to simultaneously register two projection images to a 3D image volume. The projection geometry was set up to simulate the x-ray imaging system that attached to a medical linear accelerator for cancer treatment. The normalized correlation was used as the similarity measure and the Powell's optimizer was used as the optimization method. Siddon-Jacobs fast ray-tracing algorithm was implemented to compute projection images from a 3D image volume.

Contents

1 Introduction

The intensity-based rigid 2D/3D registration has been used in the field of radiation therapy for the assessment of patient setup before daily treatment. For an imaging system that involves an x-ray imager that attached to a medical linear accelerator (linac), the projection images can be acquired at an arbitrary projection angle when the linac treatment gantry is rotating around its axis. In practice, the radiographs acquired at two (preferably orthogonal) projection angles can be used to simultaneously register with the treatment planning CT image to determine the 3D patient position. If large setup deviation from the planned position is detected, the patient can be manually or automatically adjusted before the start of the treatment. Currently, the original ITK implementation for intensity-based 2D/3D registration (e.g., **IntensityBased2D3DRegistration** in InsightApplications) that involves itk::RayCastInterpolateImageFunction only aligns one projection to a 3D image. The projection geometry is also limited and it is difficult to set up a projection with an arbitrary projection angle. In this documentation, we described our implementation of the two-projection 2D/3D registration framework that addressed above problems and presented an application of this framework to the patient setup assessment.

2 Two-Projection 2D/3D Registration Framework

The schematic plot of the proposed two-projection 2D/3D registration framework is shown in Figure 1. The following components were added to the original ITK package.

The overall process of the proposed registration method is controlled by a new class **itk::TwoProjectionImageRegistrationMethod**, which was modified from the original ITK class itk::ImageRegistrationMethod. The new class is able to handle two 2D fixed images and two interpolators.

The similarity measure is handled by a new base class **itk::TwoImageToOneImageMetric**, which was modified from the ITK class itk::ImageToImageMetric. The new class has two fixed image pointers and two interpolator pointers. Then for each particular metric, a child class should be created. In our implementation, the normalized correlation was used as the similarity measure. This measure has been shown to have very good accuracy and robustness for this application [3]. Consequently we created a child class **itk::NormalizedCorrelationTwoImageToOneImageMetric,** which was modified from the ITK class itk::NormalizedCorrelationImageToImageMetric. In particular, the member function **GetValue()** was modified. The other two member functions, **GetDerivative()** and **GetValueAndDerivation()**, that involves gradient calculation have not been implemented currently. So our new metric is limited to optimizers that do not require gradient information.

The generation of the 2D projection images from the 3D image relies on a ray-tracing interpolation function. In this work, we implemented Siddon-Jacobs fast ray-tracing algorithm [1, 2] in the new class **itk::SiddonJacobRayCastInterpolateImageFunction**.

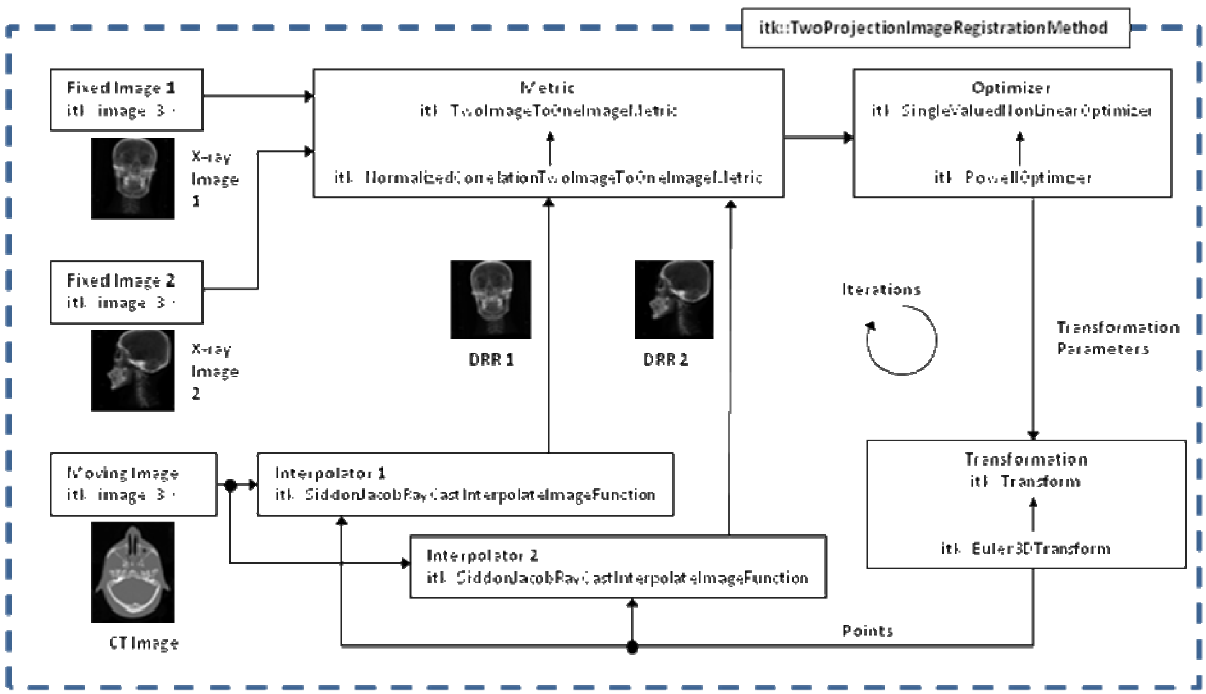

Figure 1 The schematic plot of the proposed two-projection 2D/3D registration framework.

3 Linac X-Ray Imaging Projection Geometry

The coordinate system for our application is shown in Figure 2. This system requires the input 3D image has the DICOM LPS orientation. The origin is located at the treatment isocenter. The x-ray source of the imaging system is located at either in the treatment head (for megavoltage portal imaging) or in the x-ray tube attached to the rotating gantry at 90 degree to the treatment head (for kilovoltage x-ray imaging). In both cases, the image detector (i.e., imager) planes are perpendicular to their x-ray central axes and at the opposite side of the isocenter. All x-ray sources and imagers rotate about the z-axis and the isocenter. The projection angle, the angle between the current x-ray source central axis and the minus Y-axis, was used to specify a particular projection setting. To align a 3D image with our coordinate system, the user needs to provide the continuous voxel indices of the isocenter location with respect to the image volume. The pixel spacing of the 2D projection images are specified at the plane that is perpendicular to the central axis and contains the isocenter. By default, the central axis intercepts with the center of the projection image. Otherwise, the central axis position can be given by the continuous indices as shown in Figure 3.

In order to simplify the coding for handling various projection angles, all projection settings are transform to the standard projection setting, in which the x-ray source is located at the origin, the x-ray central axis

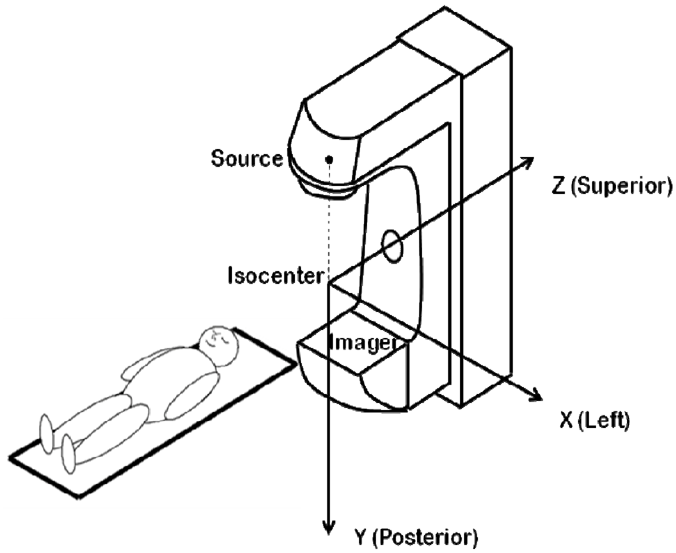

Figure 2 The imaging system and the coordinate system.

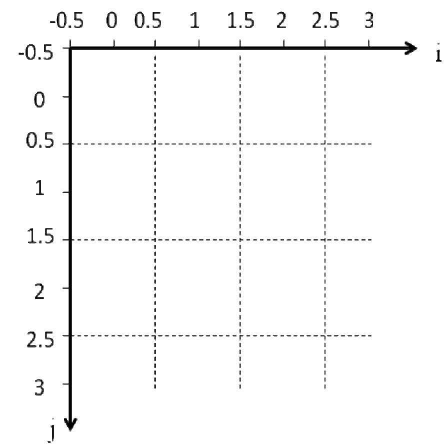

Figure 3 The 2D projection image continuous index coordinate system.

points down the negative z-axis, and the isocenter is located at the -scd (source-to-isocenter distance) on the z-axis. In our code, a composite transform was used to combine the displacement of the CT volume, the rotation of the gantry, and the transformation from an arbitrary projection angle to the standard projection setting. The following code snippet shows how this was carried out.

```
m_ComposedTransform->SetIdentity();
m_ComposedTransform->Compose(m_Transform, 0);

TransformType::InputPointType isocenter;
isocenter = m_Transform->GetCenter();
// An Euler 3D transform is used to rotate the volume to simulate the rotation of the
// linac gantry.
// The rotation is about z-axis. After the transform, a AP projection geometry
// (projecting towards positive y direction) is established.
m_GantryRotTransform->SetRotation( 0.0, 0.0, -m_ProjectionAngle );
m_GantryRotTransform->SetCenter( isocenter );
m_ComposedTransform->Compose(m_GantryRotTransform, 0);

// An Euler 3D transfrom is used to shift the source to the origin.
TransformType::OutputVectorType focalpointtranslation;
focalpointtranslation[0] = -isocenter[0];
focalpointtranslation[1] = m_scd - isocenter[1];
focalpointtranslation[2] = -isocenter[2];
m_CamShiftTransform->SetTranslation( focalpointtranslation );
m_ComposedTransform->Compose(m_CamShiftTransform, 0);
```

```
// A Euler 3D transform is used to establish the standard negative z-axis projection
// geometry. (By default, the camera is situated at the origin, points down the
// negative z-axis, and has an up-vector of (0, 1, 0).)

m_ComposedTransform->Compose(m_CamRotTransform, 0);

// The overall inverse transform is computed. The inverse transform will be used by
// the interpolation procedure.
m_ComposedTransform->GetInverse( m_InverseTransform);
```

4 Siddon-Jacob Fast Ray-Tracing Algorithm

The fast ray-tracing we implemented was first proposed by Siddon [1] and then improved by Jacobs *et al.* [2]. This algorithm calculates the exact radiological path through a voxel space. It is expected to be at least an order of magnitude faster than the linear interpolation-based ray-tracing methods.

5 Software Tests

In order to test our registration program, a DRR generation program, **getDRRSiddonJacobRayTracing**, was added. Given an input CT image, this program computes the projection image using Siddon-Jacobs ray-tracing algorithm for particular projection geometry. In our tests, we generated the projection images from two orthogonal projection angles: one at 0 degree and the other at 90 degrees, for a known CT image displacement. These two projection images were used by our 2D/3D registration application program, **2D3DTwoProjRegistration**, to register with the same CT image used to generate the projection images. Ideally, the registration output should report exactly the same displacement when the projection images were computed.

We used the CT scan of an anthropomorphic cranial phantom in our tests. The original image had 283 slices with the cross-section size of 512 x 512. The voxel physical dimension was 0.7813 x 0.7813 x 1.0 mm^3. A re-sampled version of the CT image was also included in our tests, which had the size of 200 x 200 x 142 and a voxel size of 2 x 2 x 2 mm^3. The size of the projection images (i.e., DRRs) were 512 x 512 and 256 x256 for the original and the re-sampled CT images, respectively, with their pixel sizes of 0.5 x 0.5 mm^2 and 1 x 1 mm^2, respectively. Rotations of -3, 4, and 2 degrees about the x, y, and z-axis, respectively, and translations of 5 mm to each x, y, and z-axis were applied to the CT images when generating two test projection images. The registrations were started with no rotations and translations. For the Powell's optimization method, the maximum number of line search iterations was set to 4. For both CT images, the registration terminated at 3 iterations (on top of line search iterations).

The comparison of the computation time and the registration results using the two CT images were shown in Table I. Although our tests were very limited, we found no appreciable compromising in accuracy while achieving an order of magnitude of reduction in computation time when down-sampled CT image was used. The computation was performed on a 2.53 GHz Laptop PC (Intel Core 2 Duo Processor P8700) with a Windows 7 64-bit operating system.

Table I. The comparison of the computation time and the registration results using the two CT images.

Image size		Computation time		Difference from known transform parameters					
				Rotation (deg)			Translation (mm)		
CT image	DRR	DRR generation (s)	Registration (min)	x	y	x	y	x	y
Re-sampled	256x256	0.18	1.61	-0.05	0.03	0.05	-0.06	-0.01	0.02
Original	512x512	2.15	18.97	-0.03	0.04	0.05	-0.03	0.01	-0.02

The DRRs generated using the original and the re-sampled CT images are shown in Figures 4 and 5, respectively.

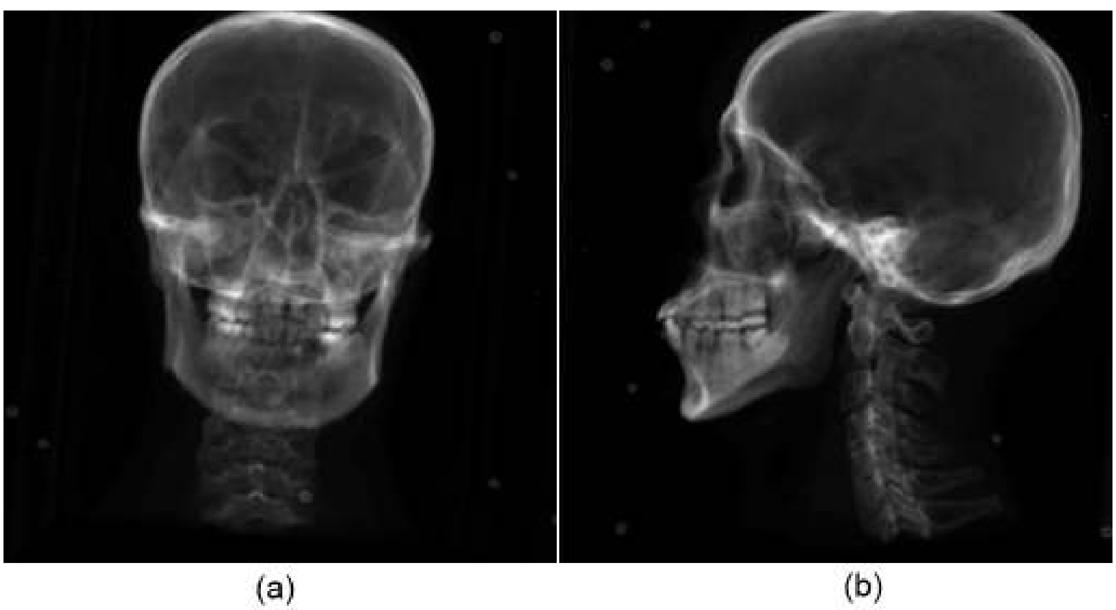

Figure 4 The DRRs generated using the original CT image. (a) Anterior-posterior view and (b) left-lateral view.

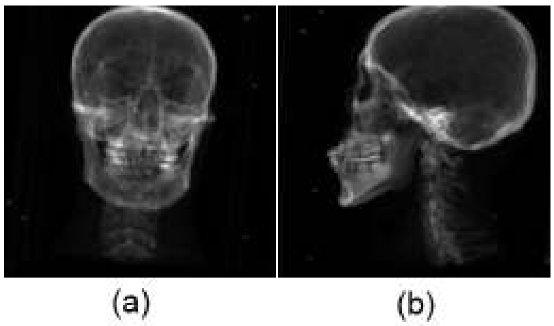

Figure 5 The DRRs generated using the re-sampled CT image. (a) Anterior-posterior view and (b) left-lateral view.

6 Software Requirements

This software package was developed on a Windows 7 64-bit computer. It has been successfully tested with ITK version 3.20.0, and CMake version 2.8.2 (Windows binary), and Microsoft Visual C++ 2008 Express Edition.

You need to have the following software installed:

- Insight Toolkit 3.20.0
- CMake 2.8.2

Note that other versions of the Insight Toolkit are also available in the testing framework of the Insight Journal. Please refer to the following page for details

http://www.insightsoftwareconsortium.org/wiki/index.php/IJ-Testing-Environment

A Additional Tools

In the software package, besides the programs described above, the following two simple programs are also included.

DicomSeriesReadNiftiImageWrite

This program reads CT slices as DICOM series and saves it as a NIFTI volumetric image. This code is provided because most of the commercial software can export CT images in DICOM format. It is worth to mention that the Analyze 7.5 image format does not support the patient orientation used in our application. So the explicit conversion to the NIFTI format was implemented.

ReadResampleWriteNifti

This program reads an image volume and save the re-sampled image in the NIFTI format. The down-sampling ratio was hard-coded. But it is very easy to modify.

B Test Commands

The following commands were used to generate the test DRRs and perform the 2D/3D registrations.

1) Test using down-sampled CT

getDRRSiddonJacobRayTracing -rp 0 -rx -3 -ry 4 -rz 2 -t 5 5 5 -iso 99.62 101.18 65 -res 1 1 -size 256 256 -o boxheadDRRDev1_G0.tif BoxheadCT.img

getDRRSiddonJacobRayTracing -rp 90 -rx -3 -ry 4 -rz 2 -t 5 5 5 -iso 99.62 101.18 65 -res 1 1 -size 256 256 -o boxheadDRRDev1_G90.tif BoxheadCT.img

2D3DTwoProjRegistration -res 1 1 1 1 -iso 99.62 101.18 65 -o boxheadDRRDev1_G0_Reg.tif boxheadDRRDev1_G90_Reg.tif boxheadDRRDev1_G0.tif 0 boxheadDRRDev1_G90.tif 90 BoxheadCT.img

2) Test using full size CT

getDRRSiddonJacobRayTracing -rp 0 -rx -3 -ry 4 -rz 2 -t 5 5 5 -iso 255 259 130 -res 0.5 0.5 -size 512 512 -o BoxheadDRRFullDev1_G0.tif BoxheadCTFull.img

getDRRSiddonJacobRayTracing -rp 90 -rx -3 -ry 4 -rz 2 -t 5 5 5 -iso 255 259 130 -res 0.5 0.5 -size 512 512 -o BoxheadDRRFullDev1_G90.tif BoxheadCTFull.img

2D3DTwoProjRegistration -res 0.5 0.5 0.5 0.5 -iso 255 259 130 -o BoxheadDRRFullDev1_G0_Reg.tif BoxheadDRRFullDev1_G90_Reg.tif BoxheadDRRFullDev1_G0.tif 0 BoxheadDRRFullDev1_G90.tif 90 BoxheadCTFull.img

Reference

[1] R. L. Siddon. Fast calculation of the exact radiological path for a three-dimensional CT array. *Med Phys,* **12(2)**:252-5, 1985.

[2] F. Jacobs, E. Sundermann, B. De Sutter, M. Christiaens, and I. Lemahieu. A fast algorithm to calculate the exact radiological path through a pixel or voxel space. *Journal of Computing and Information Technology,* **6(1)**:89-94, 1998.

[3] J. Wu, M. Kim, J. Peters, H. Chung, and S. S. Samant. Evaluation of similarity measures for use in the intensity-based rigid 2D-3D registration for patient positioning in radiotherapy. Med Phys, 36(12):5391-403, 2009.