**Matlab RUF Software**

This document describes the current implementation of the RUF software which has been originally written at Johns Hopkins University by:

* Ameet Jain
* Gouthami Chintalapani
* Siddharth Vikal
* Cho?
* Marshal master student?
* Anton Deguet

This document’s authors:

* Anton Deguet
* Nathanael Kuo

# Table of contents

1. Table of contents 1

2. Revisions 2

3. Overview 3

3.1. Application flow 3

3.2. Application screenshots 4

3.2.1. Main GUI 4

3.2.2. C-arm calibration 5

3.2.3. Seed segmentation 6

3.2.4. Tracking fiducial segmentation 7

3.2.5. Validation 8

3.3. Implementation choices 9

3.4. Directories and files 10

4. Detailed descriptions 12

4.1. Image Acquisition 12

4.1.1. General description 12

4.1.2. Specifications 13

4.1.3. Discussion 13

4.2. Calibration: Masking 13

4.2.1. General description 13

4.2.2. Specifications 13

4.2.3. Discussion 14

4.3. Calibration: Dewarping 14

4.3.1. General description 14

4.3.2. Specifications 14

4.3.3. Discussion 16

4.4. Calibration: Camera Parameter Computation 16

4.4.1. General description 16

4.4.2. Specifications 16

4.4.3. Discussion 17

4.8. Masking & Dewarping 17

4.8.1. General description 17

4.8.2. Specifications 17

4.8.3. Discussion 18

4.9. Segmentation: Fiducial (FTRAC): BBs 18

4.9.1. General description 18

4.9.2. Specifications 18

4.9.3. Discussion 19

4.10. Segmentation: Fiducial (FTRAC): Lines and Ellipses 19

4.10.1. General description 19

4.10.2. Specifications 19

4.10.3. Discussion 21

4.11. Segmentation: Seeds 21

4.11.1. General description 21

4.11.2. Specifications 21

4.11.3. Discussion 22

4.12. FTRAC Pose Estimation 22

4.12.1. General description 22

4.12.2. Specifications 22

4.13. Seed matching: MARSHAL 24

4.13.1. General description 24

4.13.2. Specifications 24

4.14. Registration to US 27

4.14.1. General description 27

4.14.2. Specifications 27

4.14.3. Discussion 27

4.15. Validation 27

4.15.1. General description 27

4.15.2. Specifications 27

4.15.3. Discussion 27

# Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Author(s) | Comments |
| 2008/10/25 | 1.0 | Anton Deguet | Initial version, incomplete |
| 2008/10/28 | 1.1 | Nathanael Kuo | Filled in detailed descriptions |
| 2008/10/29 | 1.2 | Nathanael Kuo | Filled in Directories and Files |
| 2008/10/30 | 1.3 | Anton Deguet | Added some descriptions on existing sections. ,Added registration and validation sections. Added “discussion” sub sections. |
| 2008/10/30 | 1.4 | Nathanael Kuo | Added screenshots |
| 2008/10/31 | 1.5 | Anton Deguet | Added text for screenshots |

# Overview

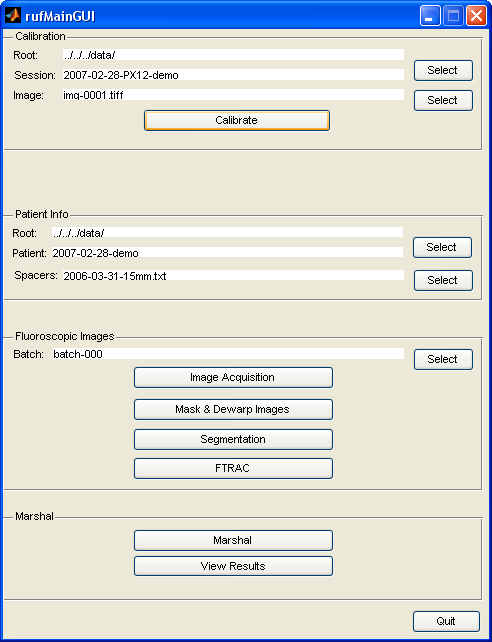
## Application flow

The overall goal of the application is to find the 3D coordinates of radioactive seeds used in prostate brachytherapy using 2D projection images obtained by a c-arm. The main steps are c-arm calibration (performed once per patient), c-arm tracking, seed reconstruction and visual validation. Reconstruction is performed at different steps during the surgery, with each reconstruction requiring a batch of images while the c-arm calibration is performed once before the surgery starts. These steps can be divided as follows:

1. c-arm calibration (performed once per patient)
   1. Image capture
   2. Image segmentation, image deformation (aka warping) features
   3. Compute image deformation
   4. Image segmentation, camera parameters
   5. Computer camera parameters
2. c-arm tracking (performed once per image)
   1. Image capture (view with seeds and tracking fiducial)
   2. Image segmentation (tracking fiducial)
   3. Pose estimation (optimization function, aka FTRAC)
3. Seed segmentation
   1. Uses image captured for c-arm tracking
   2. Image segmentation (seeds only)
4. Seed reconstruction (aka MARSHAL)
   1. Select group of images from the current batch (at least 3)
   2. Run matching algorithm
   3. Reconstruct seeds by triangulation
5. Validation
   1. Display projection of reconstructed seeds on images with highlighted segmented seeds
   2. Display simple 3D projections of reconstructed seeds along XY, XZ and YZ

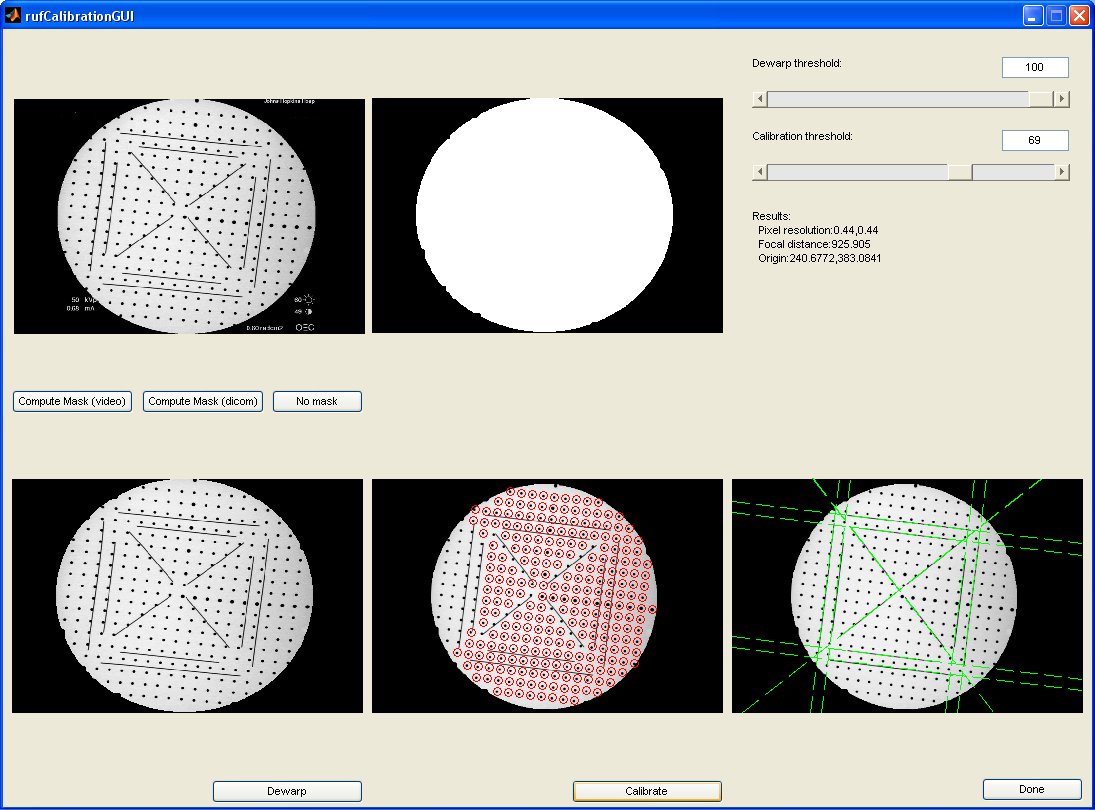
## Application screenshots

### Main GUI



The main GUI is used to define the calibration session ID, patient ID and for each patient a batch number. The application flow goes from top to bottom. “Calibration” happens only once per surgery, “Patient Info” should be entered only once per surgery and the “Fluoroscopic Images” and “Marshal” sections are used for each batch (i.e. each reconstruction).

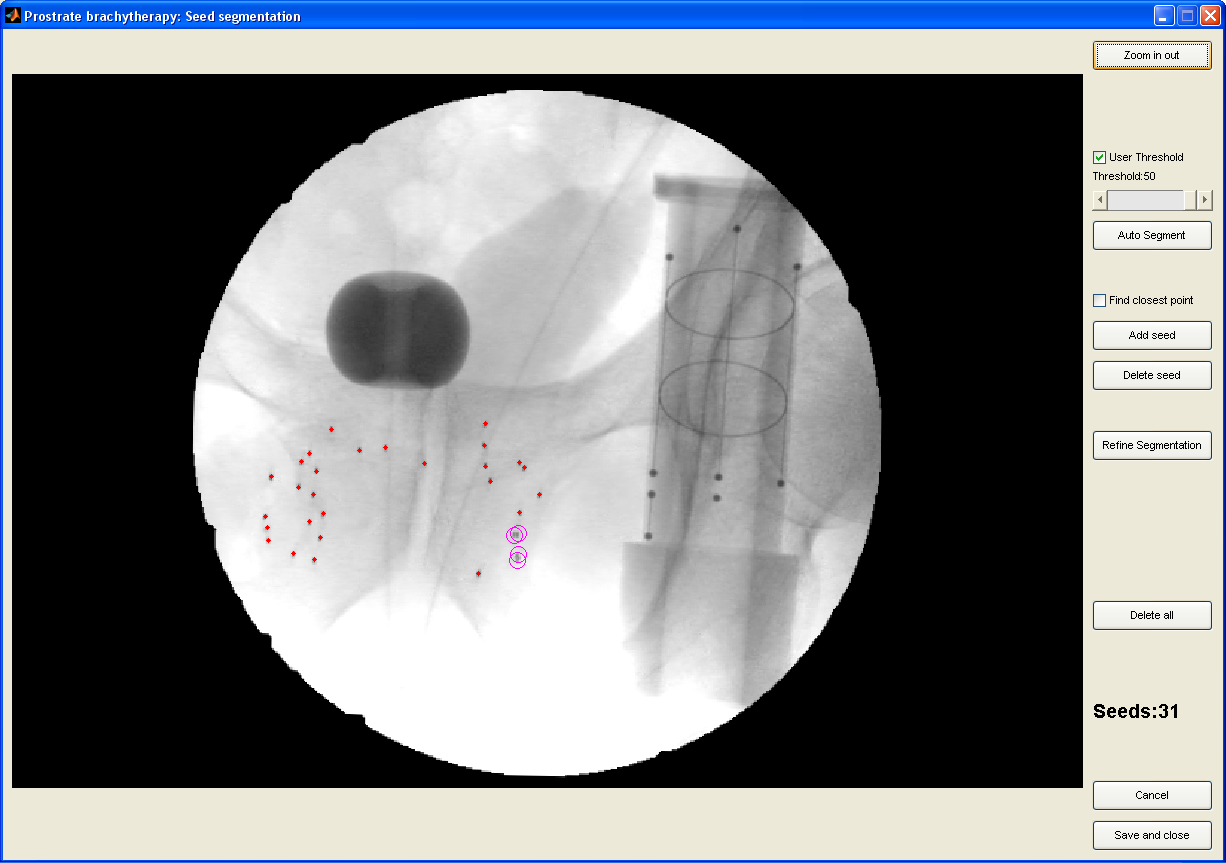
### C-arm calibration



The c-arm calibration GUI shows 5 different steps. First the original picture as captured. The second and third images are the mask computed (used to remove the extra signal outside the fluoroscopic circular image) and the original image masked. The user can then select “Dewarp”. This triggers the BB segmentation and computation of the dewarping map. The fourth image is then showed for validation. If everything goes well, the red circles are well aligned and the grid is now straight. The last step is “Calibrate”. This triggers line segmentation and line fitting. If everything goes well, all lines are showed with a green overlay and the camera parameters are displayed as text. The usual values are about 900mm focal length and the center is about 240, 320.

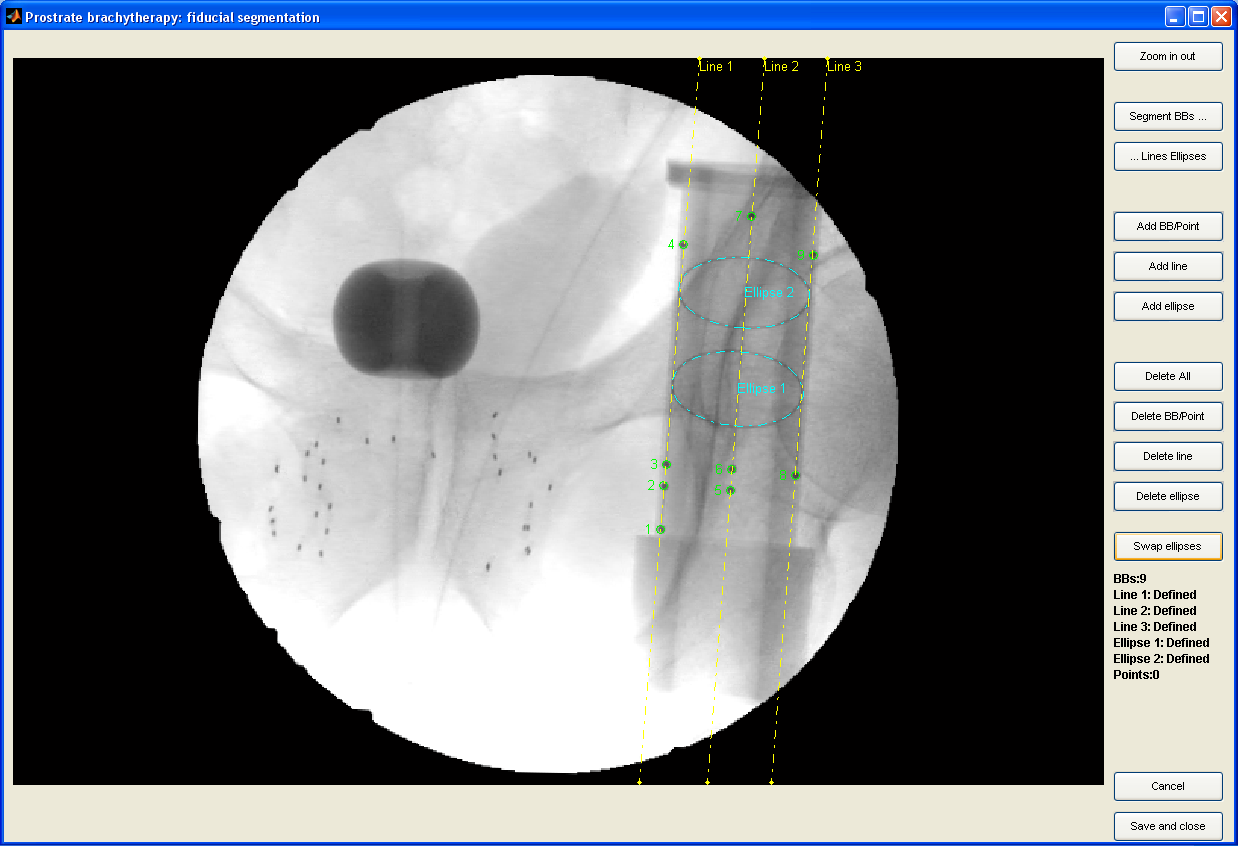
If the values and images displayed are way off, it is a clear indication that calibration went wrong. If the calibration goes wrong we can either play with the two segmentation thresholds or capture another image.

### Seed segmentation



The seed segmentation first requires selecting a region of interest. Once the ROI is selected, the user triggers the “Auto Segment” which detects most seeds automatically and labels clusters in purple. The user can then add/remove seeds manually if needed. We recently added the “Refine Segmentation” which looks at the grey levels in the region found by the morphological filters to find a better center (used 2D interpolations and uses the highest value in a 0.25 pixel resolution). Segmentation needs to be saved manually at the end. The user can come back to an existing segmentation and edit it anytime later on.

### Tracking fiducial segmentation



The tracking fiducial segmentation first requires selecting a region of interest. Once the ROI is selected, the user triggers the “Segment BBs” which uses morphological filters to segment the BBs. If this goes well, the user can then look for the lines and ellipses. If the automatic BB segmentation failed, the user can add/remove BBs manually before starting the automatic line and ellipse segmentation using “Lines and Ellipses”.

If all automatic segmentations fail, the user can perform a manual segmentation. BBs require a single click, lines require 5 clicks and ellipses require 7 clicks.

BBs must be numbered from bottom to top, left to right. Lines must be numbered from left to right and ellipses must be numbered from bottom to top. If the numbering is wrong, the next step (FTRAC) will fail, unable to find a good fit with the model.

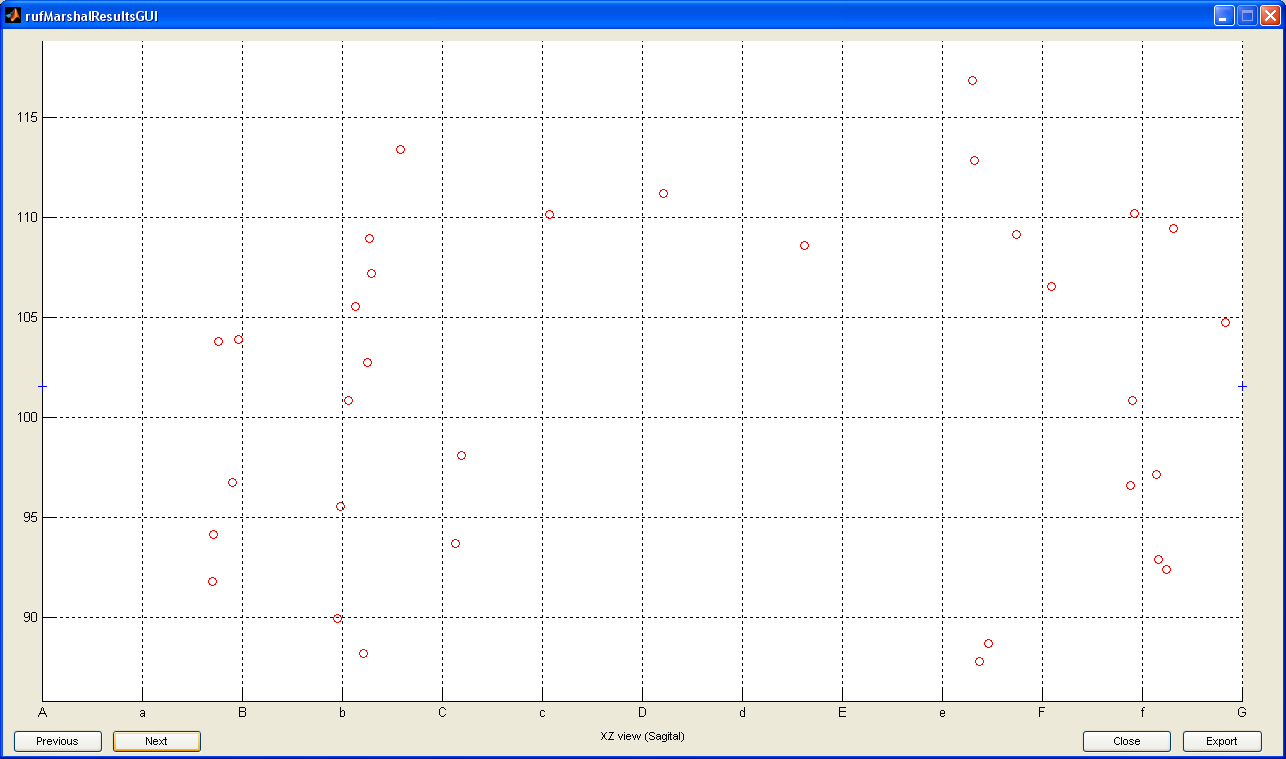
The user can come back to an existing segmentation and edit it anytime later on.

### Validation



The first images for confirmation display the fluoroscopic images used for the reconstruction. We overlay the original segmentation as well as the projection of the 3D reconstruction. We can find detect two types of issues using these views.

1. If the circles are all way off in one direction, the c-arm tracking is very likely wrong for this image. One might want to re-segment the tracking fiducial and re-run FTRAC. Please note that there will always be a small translation due to the tracking approximations.
2. If MARSHAL shows multiple circles on what seems to be a single seed, it indicates that MARSHAL found a hidden seed in the wrong place. Using multiple views is usually good to verify this assumption. The user can then go back to seed segmentation and add one seed in one image to help MARSHAL.



The last views are 2D projections of the reconstructed seeds on XY (US like, shown), XZ (c-arm like), YZ (pretty useless). The XY view is useful to find out matching errors (seeds way above or below the template) and make sure the spacers have been correctly chosen for the c-arm/US registration (Y offset of 15 or 30 mm).

## Implementation choices

The whole system is implemented in Matlab. As the code has been developed by different persons, it originally came in different directories and we preserved this organization under CVS. The directories under CVS do not correspond exactly to the different entities described previously:

* MainGUI: Glue between the other modules. Contains most of the GUI code as well as the validation part.
* Calibration: Image segmentation for the calibration phantom and computation algorithms for both image deformation and camera parameters. Does not contain the GUI associated.
* Segmentation: Tracking fiducial and seeds segmentation. This directory contains both segmentation algorithms and GUI as the segmentation is semi-automatic.
* FTRAC: Compute pose based on tracking fiducial segmentation
* MARSHAL: Using multiple seed segmentation and corresponding c-arm poses, performs the seed matching and compute the seed positions as well as their projections on the original images (used for validation).

To communicate between these different algorithms, we decided to use files. These files are automatically dated and are stored in a structured tree (dates and paths are defined by the Main GUI). This ensures that all the data is saved and allows restarting the application from any state. The tree structure always starts with two directories, “raw-data” and “processed-data”. Raw data corresponds to the data collected in the OR while processed data is the result of processing. The underlying motivation is that the raw data can be copied over to test new algorithms or implementation and one could validate the results by comparing the “processed” data.

The whole application has been initially developed to work with video images in NTSC format (8 bits grey) and short seeds (Palladium). This lead to a few hardcoded values (images are 480 by 640 and seeds are about 2mm long, i.e. about 5 pixels long). As we adapted the system for other uses, we introduced some ratios to improve the resolution independence. The first one is ratioToNTSC. This has been used to handle high resolution DICOM with 960x960 resolution. The ratio is still hard-coded but defined in one single location. For the longer seeds (Iodine) the code would require some modifications (i.e. increase morphological filters size and update the cluster classification heuristic).

The different Matlab toolbox required to run this application are:

* Image Acquisition toolbox
* Image Processing toolbox
* Statistical toolbox

## Directories and files

As mentioned earlier, the different modules communicate using files. The MainGUI functions determine the file names used and enforce a strict naming and organization. All calibrations and patients data is dated using yyyy-mm-dd. The folder “data” contains both “calibrations” and “patients”. As mentioned earlier, we the separate “raw-data” from “generated/processed-data”. Raw data is the result of data collection without any processing and can be re-used to test new implementation against existing results (generated/processed). The overall tree is described below:

* /data
  + /calibrations
    - /<session-Id> (list of calibration sessions)
      * /generated-data
        + <calib-image>-masked.tiff

Mask image of calibration image

* + - * + <calib-image>-dewarped.tiff

Masked and dewarped calibration image

* + - * + <calib-image>-map-coords.txt

Total number of mapped coordinates

List of mapped coordinates

* + - * + <calib-image>-poly-coeff.txt

Degree of Bernstein polynomial used

Size of polynomial coefficient matrix

Polynomial coefficient matrix

* + - * + <calib-image>-focal-length.txt

Focal length value

* + - * + <calib-image>-pixel-size.txt

Pixel resolution in x direction

Pixel resolution in y direction

* + - * + <calib-image>-origin.txt

x coordinate of the optical center

y coordinate of the optical center

* + - * /raw-data/video
        + <calib-image> (list of calibration images)
  + /patients
    - <patient-Id> (list of patients)
      * /processed-data
        + /intra-op/fluoro/video/

<batch-number> (list of batches starting from 000)

/3D\_Seed\_Locations (\* = ‘batch-number>\_<patient-image-1>\_<patient-image-2>-<patient\_image-3>…’)

Seeds\_\*.input

Names of the images inputted to MARSHAL

Seeds\_\*.txt

3D seed coordinates outputted from MARSHAL

Seeds\_\*\_bpe\_<patient-image>-dewarp.txt

Back projected 2D seed coordinates

/Carm\_Poses

Pose\_<patient-image>-dewarp.txt

Pose estimation matrix

Residual\_Error\_<patient-image>-dewarp.txt

Residual error values

/FTRAC\_Segmentation

BBs\_<patient-image>-dewarp.txt

Coordinates of the 9 BBs of FTRAC

Line1\_<patient-image>-dewarp.txt

Equation of segmented line 1 of FTRAC (x component of normal unit vector, y component of normal unit vector, distance from origin to line)

Line2\_<patient-image>-dewarp.txt

Equation of segmented line 2 of FTRAC (x component of normal unit vector, y component of normal unit vector, distance from origin to line)

Line3\_<patient-image>-dewarp.txt

Equation of segmented line 3 of FTRAC (x component of normal unit vector, y component of normal unit vector, distance from origin to line)

Ellipse1\_<patient-image>-dewarp.txt

Equation of segmented ellipse 1 of FTRAC

Ellipse2\_<patient-imgae>-dewarp.txt

Equation of segmented ellipse 2 of FTRAC

/Seed\_Segmentation

Data for segmented seeds(x coordinate of seed, y coordinate of seed, orientation of seed in degrees, validity of orientation of seed (0 unvalid, 1 valid), part of cluster (0 no, 1 yes))

<patient-image>-masked.tiff

Masked patient image

<patient-image>-dewarp.tiff

Dewarped patient image

* + - * /raw-data
        + /intra-op/fluoro/video/

<batch-number> (list of batches starting from 000)

<patient-image> (list of patient images)

* + - * spacerUsed.txt
        + Filename to find file for FTRAC to template transformation
* /ruf (Matlab code. The CVS repository contains more directories but we are only showing those containing code currently used).
  + /matlab
    - /Clinical\_System
    - /Dewarping
      * CalibrationLineEndPts.txt
      * PhantomOptotrakRegistrationTransformation.txt
    - /FTRAC
    - /Main\_GUI
      * /FTRAC2Template
        + <spacer>.txt
    - /Marshal\_Hidden\_Int
    - /Registration/FTRAC\_to\_Template
    - /Segmentation

# Detailed descriptions

## Image Acquisition

### General description

This module allows capturing images from a firewire based frame grabber. It uses the built-in Matlab functions which handles all the low level details. Every captured image is saved as uncompressed tiff. For recent application, we added some basic support for Dicom images. The application uses the Matlab built-in Dicom reader and converts the image to tiff.

### Specifications

1. Function name and file:
2. Inputs:
3. Outputs:
4. Dependencies:
   1. Required Matlab built-in routines:
   2. RUF function tree:

### Discussion

This module should not be ported to RadVision as RadVision already supports video capture. One should think of Dicom support, possibly using existing Dicom libraries such as DCMTK.

## Calibration: Masking

### General description

The calibration phantom image typically contains non-image data outside the field of view. The image as captured via video capture is a rectangle while the area of interest is basically a circle. The data outside the circle, if handled with the rest of the image tends to make automatic thresholding hard to do. We therefore need to compute a mask to remove the data outside the circle. The application supports three types of masks:

1. No mask
2. Dicom mask: Simple circle centered on the image
3. Video mask: Uses filters to find the large circle inside and remove the rest

### Specifications

1. Function name and file: matlab/Main\_GUI/rufCalibrationGUI.m:ComputeMask
2. Inputs:
   1. Calibration phantom image. 2D array, any precision since ComputeMask converts to double precision. Loaded through imread by rufCalibrationGUI.m from data/calibrations/<session-Id>/raw-data/video/<calib-image>
   2. Masking mode. Integer indicating 1: compute video mask, 2: compute DICOM mask, 3: no mask. Indicated in rufCalibrationGUI user interface
   3. Resolution ratio to NTSC. Double (typical value 1). Hard-coded in rufCalibrationGUI to be width of image divided by 480
3. Outputs:
   1. Mask image. 2D array, double (0-1). Saved through imwrite to data/calibrations/<session-Id>/generated-data/<calib-image>-mask.tiff
   2. Masked calibration phantom image. 2D array, uint8 (0-255)
   3. Morphologically processed masked calibration phantom image. 2D array, uint8 (0-255)
   4. Automatic dewarping threshold. Integer between 0 and 255
4. Dependencies:
   1. Required Matlab built-in routines:
      1. Image processing toolbox: imshow.m, bwlabel.m, strel.m, imopen.m, imerode.m, imcomplement.m, imsubtract.m
      2. Java method: reshape
   2. RUF function tree:
      1. Generate\_Image\_Mask
      2. Dewarping/PerformMorphologicalOperations.m

### Discussion

This module should be ported to RadVision since it is needed for the c-arm calibration.

## Calibration: Dewarping

### General description

The c-arm calibration phantom is composed of two plates. The plate closer to the image intensifier is used for distortion correction. The plate contains a regular pattern composed of metal BBs disposed on a regular grid with a 10 mm spacing. The deformation of the pattern can then be used to correct the distortion of the images. The first problem is to segment the image to identify and locate the metal BBs. This is performed using morphological filters. Once the BBs have been segmented, we use Bernstein polynomials to determine the image distortion. The result is both the polynomial coefficients and a distortion correction map.

### Specifications

1. Function name and file: matlab/Dewarping/ComputeDistortionMap.m
2. Inputs:
   1. Image of the masked calibration phantom. 2D array, uint8 (0-255). Computed in Main\_GUI/rufCalibrationGUI.m:ComputeMask
   2. Morphologically processed masked image. 2D array, uint8 (0-255). Computed in Main\_GUI/rufCalibrationGUI.m:ComputeMask
   3. Segmentation threshold. Integer between 0 and 255 (typical value 40). Automatic threshold computed in Main\_GUI/rufCalibrationGUI.m:ComputeMask or manual threshold indicated in Main\_GUI/rufCalibrationGUI user interface
   4. Resolution ratio to NTSC. Double (typical value 1). Hard-coded in rufCalibrationGUI to be width of image divided by 480
   5. Pixel resolution. [1x2 double]. The distortion allows to set any desired resolution. We could use the resolution of the initial image or set one. In the current implementation (2008/10/25), the resolution is hard coded to 0.44, 0.44 (mm per pixel).
3. Outputs:
   1. Distortion corrected image. 2D array, double (0-1). Saved through imwrite by Main\_GUI/rufCalibrationGUI to data/calibrations/<session-Id>/generated-data/<calib-image>-dewarped.tiff
   2. Map Coordinates. Saved through WriteMappedCoordsToFile by Main\_GUI/rufCalibrationGUI to data/calibrations/<session-Id>/generated-data/<calib-image>-map-coords.txt
      1. Mapping of x pixels from distorted to undistorted image. 2D array, double
      2. Mapping of y pixels from distorted to undistorted image. 2D array, double
   3. Polynomial coefficients. Saved through WritePolyCoeffToFile by Main\_GUI/rufCalibrationGUI to data/calibrations/<session-Id>/generated-data/<calib-image>-poly-coeff.txt
      1. Degree of Bernstein polynomial. Integer
      2. Polynomial coefficients. Mx2 array, double
   4. Orientation of the grid. Integer
   5. Location of the center BB in pixel units. [1x2 double]
   6. Locations of the BB centers. Mx2 array, double
   7. Pixel resolution. [1x2 double]. Saved through fprintf by Main\_GUI/rufCalibrationGUI to data/calibrations/<session-Id>/generated-data/<calib-image>-pixel-size.txt
4. Dependencies:
   1. Required Matlab built-in routines:
      1. Image processing toolbox: im2double.m, imclearborder.m, bwlabel.m, regionprops.m
      2. Java method: reshape
      3. Unknown: syms, solve
   2. RUF function tree:
      1. ProcessImage
         1. CalculateCentroid
      2. CalculateCentroid
      3. FindEntropy
      4. ComputePixelSize
      5. FindCorrespondences
         1. LabelGridPointsAroundOrigin
      6. NormalizetoImageBox
      7. ComputeBernsteinCoefficients
         1. ComputeBezierTerms
      8. DewarpImage
         1. NormalizeToImageBox
         2. ComputeBezierTerms

### Discussion

In the long run, we will need to find a practical solution for c-arm calibration, Flat panel detectors would be ideal but not economically feasible. One could also rely on the manufacturer calibration tools but this could lead to integration issues. We currently have a specific design which will only fit 9” image intensifier or 12” with magnification. Ideally, we should have a design that would fit more c-arms.

## Calibration: Camera Parameter Computation

### General description

Computes the intrinsic camera parameters of the X-ray source.

### Specifications

1. Function name and file: matlab/Dewarping/ComputeCalibrationParameters.m
2. Inputs:
   1. Dewarped image of the calibration phantom. 2D array, double (0-1). Computed in ComputeDistortionMap
   2. Pixel resolution. [1x2 double]. The distortion allows to set any desired resolution. We could use the resolution of the initial image or set one. In the current implementation (2008/10/25), the resolution is hard coded to 0.44, 0.44 (mm per pixel)
   3. Orientation of the dewarp grid. Integer. Computed in ComputeDistortionMap
   4. Location of the center BB. [1x2 double]. Computed in ComputeDistortionMap
   5. Locations of the BB centers. Mx2 array, double. Computed in ComputeDistortionMap
   6. Threshold. Integer between 0 and 1. Automatic threshold computed in Main\_GUI/rufCalibrationGUI.m or manual threshold indicated in Main\_GUI/rufCalibrationGUI user interface
   7. Resolution ratio to NTSC. Double (typical value 1). Hard-coded in rufCalibrationGUI to be width of image divided by 480
3. Outputs:
   1. Focal length of the X-ray optical system in mm. Double. Saved through fprintf by Main\_GUI/rufCalibrationGUI to data/calibrations/<session-Id>/generated-data/<calib-image>-focal-length.txt
   2. (x,y) coordinates of the optical center in pixels and z coordinate represents the focal length in mm. [3x1 double]. Saved through fprintf by Main\_GUI/rufCalibrationGUI to data/calibrations/<session-Id>/generated-data/<calib-image>-origin.txt
   3. Line end points. [12x4 double]
4. Dependencies:
   1. Required Matlab built-in routines:
      1. Image processing toolbox: imclearborder.m, bwlabel.m, imcomplement.m, strel.m, imopen.m, imsubtract.m, regionprops.m
   2. RUF function tree:
      1. ExtractLines
      2. ComputeCalibrationLineEndPoints
         1. Segmentation/least\_squares\_line.m
      3. ComputeTransformationFromGridToImage
         1. CalculateCentroid
      4. ReadPtsFromFile
      5. ProjectPtsOnGrid
      6. GenerateCostMatrix
      7. ReadPhantomRegistrationData
      8. Segmentation/least\_squares\_line.m
      9. Marshal\_Hidden\_Int/hungarian.m
      10. Registration/FTRAC\_to\_Template/fit\_least\_squares\_3dline.m
      11. Registration/FTRAC\_to\_Template/fit\_least\_squares\_plane.m

### Discussion

See discussion in previous section

## Masking & Dewarping

### General description

All images taken from patient must be masked and dewarped. The mask used is the one computed during the c-arm calibration. Dewarping is performed using the deformation map. This requires to store the map which is a fairly large data set but it is much faster than re-computing the map from the polynomial every time we need it.

### Specifications

1. Function name and file: matlab/Main\_GUI/rufDewarpGUI.m:DewarpButton\_Callback
2. Inputs:
   1. Input directory. String. Default as data/patients/<patient-Id>/raw-data/intra-op/fluoro/video/<batch-number>/
   2. Output directory. String. Default as data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/
   3. Coefficients file name. String. Default as data/calibrations/<session-Id>/generated-data/<calib-image>-map-coords.txt
   4. Mask image. 2D array, double. Computed from image loaded through imread by Main\_GUI/rufDewarpGUI from data/calibrations/<session-Id>/generated-data/<calib-image>-mask.tiff
3. Outputs:
   1. Masked image. Saved through imwrite to <input directory>/<patient-image>-masked.tiff
   2. Dewarped image. Saved through imwrite to <input directory>/<patient-image>-dewarp.tiff
4. Dependencies:
   1. Required Matlab built-in routines:
      1. Image processing toolbox: imshow.m
      2. Java method: reshape
   2. RUF function tree:
      1. Dewarping/ReadMappedCoordsFromFile.m
      2. Dewarping/DewarpImageFromMappedCoordinatesLoaded.m

### Discussion

This module should not pose problem to port.

## Segmentation: Fiducial (FTRAC): BBs

### General description

All patient images have the FTRAC in view. Lines, ellipses, and BBs must be segmented for coming C-arm tracking pose estimation step. The BBs are extracted using morphological filters within a region of interest defined by the user.

### Specifications

1. Function name and file: matlab/Segmentation/FiducialBBSegmentation.m
2. Inputs:
   1. Region of interest of masked and dewarped patient image. 2D array, uint8. Loaded through imread by Main\_GUI/rufSegmentationSelectionGUI.m from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/<patient-image>-dewarp.tiff and region of interest defined by user in FiducialSegmentationGUI user interface
   2. Resolution ratio to NTSC. Double (typical value 1). Hard-coded in rufCalibrationGUI to be width of image divided by 480
3. Outputs:
   1. BB locations array in the format [CGx1 CGy1; CGx2 CGy2; …]. [9x2 double]. Saved through save by Main\_GUI/rufSegmentationSelectionGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/BBs\_<patient-image>-dewarp.txt
4. Dependencies:
   1. Required Matlab built-in routines:
      1. Image processing toolbox: imcomplement.m, imopen.m, strel.m, imsubtract.m, bwareaopen.m, imclearborder.m, bwlabel.m, regionprops.m
   2. RUF function tree: no other function calls

### Discussion

This module is working fairly well but could be much nicer if some of the a-priori knowledge of the FTRAC design were taken into account. In the long run, it looks like the c-arm tracking will not rely on this specific tracking fiducial so it is hard to justify further development. This comment applies to the next module as well.

To note, we believe that there is already a competing C/C++ implementation at Hopkins which might actually be better.

## Segmentation: Fiducial (FTRAC): Lines and Ellipses

### General description

All patient images have the FTRAC in view. Lines, ellipses, and BBs must be segmented for coming C-arm tracking pose estimation step. See section above for region of interest. Lines are segmented using Hough transforms. Ellipses are segmented using whatever is not yet indentified (i.e. mask BBs and lines) and a voting mechanism. This approach is far from optimal as it doesn’t use the tracking fiducial model.

### Specifications

1. Function name and file: matlab/Segmentation/FiducialSegmentationGUI.m
2. Inputs:
   1. Region of interest. [1x1 struct]; .StartX, .StartY, .SizeX, .SizeY integers; defined by user in FiducialSegmentationGUI user interface
   2. BB locations. [9x2 double]. Computed in FiducialBBSegmentation
   3. Masked and dewarped patient image. 2D array, uint8. Loaded through imread by Main\_GUI/rufSegmentationSelectionGUI.m from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/<patient-image>-dewarp.tiff
   4. Resolution ratio to NTSC. Double (typical value 1). Hard-coded in rufCalibrationGUI to be width of image divided by 480
3. Outputs:
   1. Line 1. [1x3 double]. Saved through save by Main\_GUI/rufSegmentationSelectionGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Line1\_<patient-image>-dewarp.txt
   2. Line 2. [1x3 double]. Saved through save by Main\_GUI/rufSegmentationSelectionGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Line2\_<patient-image>-dewarp.txt
   3. Line 3. [1x3 double]. Saved through save by Main\_GUI/rufSegmentationSelectionGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Line3\_<patient-image>-dewarp.txt
   4. Ellipse 1. [1x6 double]. Saved through save by Main\_GUI/rufSegmentationSelectionGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Ellipse1\_<patient-image>-dewarp.txt
   5. Ellipse 2. [1x6 double]. Saved through save by Main\_GUI/rufSegmentationSelectionGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Ellipse2\_<patient-image>-dewarp.txt
4. Dependencies:
   1. Required Matlab built-in routines:
      1. Image processing toolbox: imcomplement.m, imclearborder.m, imclose.m, strel.m, bwlabel.m, regionprops.m, imtophat.m, bwareaopen.m
   2. RUF function tree:
      1. BuildEdgeImage
      2. GetLineStartAndEndPointsFromBBs
      3. ExtractLine
         1. FindThresholds
         2. LocateUpConnectedPoint
            1. IsWithinIntensityRange
         3. LocateDownConnectedPoint
            1. IsWithinIntensityRange
         4. GetEdgeByGraphSearch
            1. GetEdgeByGraphSearchTraverseUpBottom

IsWithinIntensityRange

FindDirectionAssociatedCost

FindDistance

IsNodeInTheList

FindLowestCostNode

* + - * 1. GetEdgeByGraphSearchTraverseBottomUp

IsWithinIntensityRange

FindDirectionAssociatedCost

FindDistance

IsNodeInTheList

FindLowestCostNode

* + - 1. FindGoodPoint
      2. least\_squares\_line
    1. CalculateLineDirectionEstimateFromEndPoints
    2. RecalculateBBLocationsAndEstablishCorrespondence
       1. IsOnLine
       2. FindDistance
    3. NewExtractEllipses
       1. MorphoReducedImage
       2. EllipseEqFromPoints
       3. EllipseParamFromEq
       4. IsCenterPresentInExistingWindows
       5. AddtoExistingWindow
       6. DefineNewWindow
          1. ComputeRegionEnclosingCenter
    4. EllipseTranslate
    5. DisplayFiducialSegmentation
       1. EllipseTranslate
       2. EllipseParamFromEq
       3. EllipsePointsFromParam

### Discussion

See previous module.

## Segmentation: Seeds

### General description

Seeds must be segmented for coming MARSHAL step. The seed segmentation is a three steps process:

1. Seed segmentation using morphological filters. Size of filters depends on the image resolution and type of seeds.
2. Cluster classification. If a cluster of seeds is detected, try to break it apart and guess the number of seeds in the cluster.
3. Segmentation refining. Use grey levels in the area found by the morphological filters to find a sub-pixel position (in .25 pixels)

### Specifications

1. Function name and file: matlab/Segmentation/BrachytherapySeedSegmentation.m
2. Inputs:
   1. Region of interest of masked and dewarped patient image. 2D array, uint8. Loaded through imread by Main\_GUI/rufSegmentationSelectionGUI.m from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/<patient-image>-dewarp.tiff and region of interest defined by user in SeedSegmentationGUI user interface
   2. Threshold flag. Boolean. If true, uses manual threshold; if false, uses automatic threshold
   3. Manual threshold. Integer between 10 and 50.
   4. Resolution ratio to NTSC. Double (typical value 1). Hard-coded in rufCalibrationGUI to be width of image divided by 480
3. Outputs:
   1. Final data set in form of [CGx CGy Orientation OrientationValidity PartOfCluster]; Mx5 double; Saved through save by Main\_GUI/rufSegmentationSelectionGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/Seed\_Segmentation/Seeds\_<patient-image>-dewarp.txt
   2. Current threshold. Double
4. Dependencies:
   1. Required Matlab built-in routines:
      1. Image processing toolbox: imcomplement.m, imopen.m, strel.m, imsubtract.m, imclearborder.m, bwlabel.m, regionprops.m, bwperim.m, bwmorph.m, imhist.m
      2. Statistics toolbox: prctile.m
      3. Java method: reshape
   2. RUF function tree:
      1. FindEntropyThreshold
      2. CalculateLengthStats
      3. CalculateAreaStats
      4. CalculatePerimeterStats
      5. CalculateWidthStats

### Discussion

This module works fairly well. The two issues that would need to be addressed for a generalized application are morphological filters for iodine seeds as well as cluster separation for iodine seeds. For research purposes, we might also want to work on seed orientation which could help solve the matching problem.

## FTRAC Pose Estimation

### General description

Estimated pose for C-arm tracking must be computed from the segmented FTRAC and intrinsic camera parameters for coming seed matching step. This uses a brute force optimization. The result is a pose as well as a residual error used to estimate the quality of the pose.

### Specifications

1. Function name and file: matlab/Clinical\_System/evaluate\_pose\_ftrac.m
2. Inputs:
   1. BBs. [9x2 double]. Loaded through load by Main\_GUI/rufMainGUI from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/BBs\_<patient-image>-dewarp.txt
   2. Line 1. [1x3 double]. Loaded through load by Main\_GUI/rufMainGUI from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Line1\_<patient-image>-dewarp.txt
   3. Line 2. [1x3 double]. Loaded through load by Main\_GUI/rufMainGUI from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Line2\_<patient-image>-dewarp.txt
   4. Line 3. [1x3 double]. Loaded through load by Main\_GUI/rufMainGUI from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Line3\_<patient-image>-dewarp.txt
   5. Ellipse 1. [1x6 double]. Loaded through load by Main\_GUI/rufMainGUI from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Ellipse1\_<patient-image>-dewarp.txt
   6. Ellipse 2. [1x6 double]. Loaded through load by Main\_GUI/rufMainGUI from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/FTRAC\_Segmentation/Ellipse2\_<patient-image>-dewarp.txt
   7. Pixel ratio. [1x2 double]. Computed from pixel resolution, which was loaded through fscanf by Main\_GUI/rufMainGUI from data/calibrations/<session-Id>/generated-data/<calib-image>-pixel-size.txt
   8. Screen size. [0x0 double]
   9. Origin. [1x2 double]. Loaded through fscanf by Main\_GUI/rufMainGUI from data/calibrations/<session-Id>/generated-data/<calib-image>-origin.txt
   10. Focal length. Double. Loaded through fscanf by Main\_GUI/rufMainGUI from data/calibrations/<session-Id>/generated-data/<calib-image>-focal-length.txt
   11. Initial pose estimate. [4x4 double]. Hard coded to be [0 0 -1 0; 1 0 0 0; 0 -1 0 700; 0 0 0 1]
3. Outputs:
   1. Pose. [4x4 double]. Saved through save by Main\_GUI/rufMainGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/Carm\_poses/Pose\_<patient-image>-dewarp.txt
   2. Residual error. Mx1 array, double. Saved through save by Main\_GUI/rufMainGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/Carm\_poses/Residual\_Error\_<patient-image>-dewarp.txt
4. Dependencies:
   1. Required Matlab built-in routines:
      1. Image processing toolbox: fspecial.m, imshow.m
   2. RUF function tree:
      1. FTRAC/Fiducial\_points.m
         1. ellipse\_points
            1. xyz\_xform
      2. FTRAC/Estimate\_Pose.m
         1. Ellipse\_check
         2. xyz\_form
         3. Find\_Solution
            1. Next\_Pose

Get\_jacobian

get\_projected\_pixels

xyz\_xform

* + - 1. Is\_Converged
      2. Choose\_Transform

## Seed matching: MARSHAL

### General description

This step takes three or more images and matches the segmented seeds among the images.

### Specifications

1. Function name and file: matlab/Marshal\_Hidden\_Int/marshal.m
2. Inputs:
   1. Segmented seeds. [Mx1 cell array], each cell [Mx2 double]. Loaded through load by Main\_GUI/rufMarshalSelectionGUI from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/Seed\_Segmentation/Seeds\_<patient-image>-dewarp.txt for selected patient images
   2. Camera parameters. [Mx1 cell array], each cell [1x1 struct], each struct .ratio [2x1 double], .origin [2x1 double], .focal [double]. Loaded through fscanf by Main\_GUI/rufMarshalSelectionGUI from data/calibrations/<session-Id>/generated-data/<calib-image>-pixel-size.txt, <calib-image>-origin.txt, <calib-image>-focal-length.txt
   3. Poses. [Mx1 cell array], each cell [4x4 double]. Loaded through load by Main\_GUI/rufMarshalSelectionGUI from data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/Carm\_poses/Pose\_<patient-image>-dewarp.txt for selected patient images
   4. Total seeds. Integer. Maximum of number of segmented seeds among selected patient images
3. Outputs:
   1. 3D seed coordinates. [Mx3 double]. Saved through save by Main\_GUI/rufMarshalSelectionGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/3D\_Seed\_Locations/Seeds\_<batch-number>\_<patient-image-1>\_<patient-image-2>-<patient\_image-3>….txt after making transformation to FTRAC template
   2. BPE (back projection error?). [Mx1 double]
   3. 2D seed back projections. [1xN cell array], each cell [Mx2 double]. Saved through save by Main\_GUI/rufMarshalSelectionGUI to data/patients/<patient-Id>/processed-data/intra-op/fluoro/video/<batch-number>/3D\_Seed\_Locations/Seeds\_<batch-number>\_<patient-image-1>\_<patient-image-2>-<patient\_image-3>…\_bpe\_<patient-image-#>-dewarp.txt
   4. Seed matchings. Shows corresponding segmented seed numbers. [Mx3 double]
   5. Seed matching image indices. Shows indices of the images used for seed matchings (useful when number of images is greater than 3). [1x3 double]
4. Dependencies:
   1. Required Matlab built-in routines:
      1. Java method: reshape
   2. RUF function tree:
      1. compute\_matchings
         1. generate\_cost\_matrices
            1. generate\_2img\_cost\_matrix

sm\_recon\_general\_2

sm\_backproj\_general

* + - 1. create\_node\_edges
      2. create\_initial\_flow
         1. hungarian
      3. create\_total\_network
      4. match\_cycle
         1. match\_bellmanford\_matlab
      5. extract\_flow\_matching
      6. process\_matching\_flow
      7. find\_loops
         1. remove\_subgroups

calculate\_bpe

sm\_recon\_general\_2

sm\_backproj\_general

* + - * 1. current\_cost
      1. process\_image\_matching
         1. resolve\_loops\_one

sm\_comb\_all

sm\_comb\_all\_cost2

sm\_recon\_core2

sm\_combset\_match

sm\_combset\_all

sm\_combset\_match\_bypart

break\_down\_loops

generate\_cost\_matrices (see 4bi1)

create\_node\_edges

create\_initial\_flow (see 4bi3)

create\_total\_network

match\_cycle (see 4bi5)

extract\_flow\_matching

process\_matching\_flow

find\_loops (see 4bi8)

resolve\_outer\_hidden\_seeds

generate\_cost\_matrices (see 4bi1)

create\_node\_edges

create\_initial\_flow (see 4bi3)

create\_total\_network

match\_cycle (see 4bi5)

extract\_flow\_matching

process\_matching\_flow

find\_loops\_outer

remove\_subgroups (see 4bi8a)

resolve\_loops\_special

sm\_comb\_all

sm\_comb\_all\_cost2 (see 4bi9aii)

sm\_combset\_match (see 4bi9aiii)

sm\_combset\_match\_bypart

break\_down\_loops (see 4bi9av)

calculate\_bpe (see 4bi8ai)

sm\_recon\_general\_2

* + - * 1. resolve\_outer\_hidden\_seeds

generate\_cost\_matrices (see 4bi1)

create\_node\_edges

create\_initial\_flow (see 4bi3)

create\_total\_network

match\_cycle (see 4bi5)

extract\_flow\_matching

process\_matching\_flow

find\_loops\_outer (see 4bi9av9h)

* + - * 1. resolve\_loops

sm\_comb\_all

sm\_comb\_all\_cost2 (see 4bi9aii)

sm\_combset\_match (see 4bi9aiii)

sm\_combset\_match\_bypart

break\_down\_loops (see 4bi9av)

sm\_recon\_general\_2

* + - * 1. calculate\_bpe (see 4bi8ai)
    1. calculate\_bpe\_final
       1. sm\_recon\_general\_2
       2. sm\_backproj\_general

## Registration to US

### General description

The seeds reconstructed by MARSHAL are defined in the FTRAC coordinate system. In the current implementation, FTRAC is rigidly attached to the needle template. We designed both the tracking fiducial and the needle template with features that can be located using an Optotrak tool (grooves).

To handle different patients (thin to large), we added spacers (0mm, 15mm and 30mm) which are placed between the template and the tracking fiducial. In general, we try to use no spacer and keep the tracking fiducial as close as possible to the prostate and avoid problems related to the limited field of view. For larger patients, we need to add spacers to prevent the tracking fiducial from being pushed.

### Specifications

In the current implementation, the transformations corresponding to each spacersare saved in text files and we have to specify which one is used via the GUI.

### Discussion

The US/Fluoro registration is one the weak points of the system. We should probably not invest too much time working on the existing approach.

## Validation

### General description

Once the reconstruction has been computed, we need some basic validation tools. We decided to provide two different views:

1. On each c-arm image used for reconstruction, display the initial segmentation as well as the projection of the 3D reconstruction. This is useful to figure out what MARSHAL considers to be a hidden seed or cluster. In an interactive mode, the user can go back to the image segmentation and add seeds manually to help MARSHAL identify hidden seeds/clusters.
2. A synthetic view of the seeds with an overlay of the needle template along three directions, XY (facing template), XZ (looks like c-arm view) and YZ (not very useful). The reconstruction tends to place seeds too high or too low if the matching fails. In the first view, these incorrect seeds shown either below or above the template which is obviously impossible.

### Specifications

### Discussion

This code should probably be rewritten to take advantage of all the extra information available in RadVision (i.e. US images, prostate segmentation, plan. marked seeds).