Documentation WFS-SIM V3

David Haberthür

for Rev 31 - 2008-12-23 17:24:26 +0100 (Tue, 23 Dec 2008)

This — rewritten — version of the Wide Field Scan Simulation is comprised of the files in the folder `\MATLAB\wfs-sim\v3`. The necessary files are:

- main.m
- fct_ErrorCalculation.m
- fct_InterpolateImage.m
- fct_SegmentGenerator.m

The other files in the directory are for testing purposes (hence the `test_`-prefix).

1 main.m

`main.m` is the main MATLAB-File that is used for the simulation. The user is first asked for some input parameters like desired FOV (`FOV_mm`), `Binning`, `Magnification`, Overlap between SubScans (`Overlap_px`) and settings for the Quality (`MinimalQuality`, `MaximalQuality` and `QualityStepWidth`). From the Binning we can calculate the `DetectorWith`, since the camera is 2048 s wide and we assume that the full sensor is used. The `pixelsize` can be calculated from the Magnification, according to the table on the TOMCAT website. The DetectorWidth and the Overlap define the width of one segment (`SegmentWidth_px`) and thus the `AmountOfSubScans`. If the sample does not perfectly fit in the diameter of AmountOfSubScans $\times$ SegmentWidth, we're actually "loosing" imaging real estate, thus the user is informed of this. The actual FOV (`ActualFOV_px`) is used for the calculation of the correct number of projections.

After we make sure that we have an odd amount of SubScans, we use the function `fct_SegmentGenerator` (see section 4 for details on this function) to calculate the `NumberOfProjections` from the inputs. The resulting table is X amounts of SubScans wide and Y protocols tall.

The user specifies the image size that should be used for the calculation of the simulated error (`SimulationSize_px`). Since the radon and inverse radon transform is quite processor-intensive we simulate with a model where the images and reconstructions are of a smaller size. While making sure that the overlap for this model still stays above 5 pixels, we calculate the reduction factor according to the user input (`ModelReductionFactor`) and scale the table with the number of projections, the phantom image (`ModelImage`) and the detector width (`ModelDetectorWidth`) with this reduction factor. After we have calculated a full size sinogram and a full size reconstruction (`ModelMaximalSinogram` and `ModelMaximalReconstruction`), we use the function `fct_ErrorCalculation` (see section 2 for details on this function) to calculate the error of each protocol.

This error is kept track of in `AbsoluteError` and `ErrorPerPixel(Protocol)`, once as absolute and once normalized to the total amount of pixels in the image. After we've sorted the number of projections and summed the amount of projections for each protocol (`TotalSubScans`), we plot this value versus the `QualityMeasure`. The Quality is calculated in such a way that we subtract the maximum of the Error per Pixel from all the Errors per Pixel, so we get a high value for low Errors and vice versa. After we've scaled this value to the minimal and maximal quality the user has input, we provide the user a mean of choosing the protocol, that suits his needs in terms of quality and scanning time.

The chosen value is saved into `User_NumProj` and subsequently written to disk to a file `UserSampleName`.txt which contains the details of the chosen Protocol, including `InBeamPosition` and the angles of the rotation stage (which are currently hardcoded as `RotationStartAngle`=46° and `RotationStopAngle`=225°).

2 fct_ErrorCalculation.m

This function uses the model phantom, the number of projections of each subscan and the maximal reconstruction as an input. Inside the function we split the image into different parts (no overlap is calculated!) for each subscan. According to the different amount of subscans from the outer to the inner scans we interpolate rows of these parts from the sinogram of the maximal image using another function (`fct_InterpolateImage`). This corresponds to a different amount of recorded projections, since these are "encoded" in the rows of the sinogram. Since MATLAB calculates the sinogram 90° rotated to what we know and like, the sinogram is internally flipped (thus we get the "interpolating in horizontal direction" when the numbers of projections for the scans need to be interpolated.

The resulting reconstruction from the correctly interpolated sinogram is shown in together with the difference image. An error-measure is then calculated from this image in such a way that we take the sum over each pixel — in MATLAB written as $\sum_{horiz.}(\sum_{vert.}(\text{DiffImg}))$ — of the difference image of the resulting reconstruction and the model image. The difference image is calculated with the MATLAB-function `imabsdiff` which calculates the absolute difference between two images, so we don't have to take the square of this error.

3 fct_InterpolateImage.m

This function came from my lack of memorization of the correct interpolation syntax. It takes an image as input, wants to know how many lines should be interpolated over (interpolate every x^{th} line and spits out the interpolated image. An optional parameter decides if it should interpolate horizontally or vertically (needed since MATLAB makes the sinograms 90° rotated to what we like and know...).

4 fct_SegmentGenerator.m

This function takes the sample width (`ActualFOV_px`), the amount of SubScans and the quality details as input. The fundamental number of projections N is calculated as $N = \mathrm{FOV} * \frac{\pi}{2}$ which is according to the normally recorded 1500 projections for a detector size of 1024 pixels.

Afterwards a table with the number of projections for each segment/subscan is generated. If the quality ranges from 10–100% in 10% steps we have 10 protocols. Since we want to have a reduction in time, we now divide the inner protocols with 2, 4, 6, 8 and 10. This also works if we have not three, but five or seven subscans. All the protocols where the sampling rate is lower than 30% are discarded (but only if the minimal quality specified by the user ain't lower than that...).

In the end we get a table with X colums and Y rows, where X is the amount of subscans and Y are the different protocols.