Version 1.2

How to use

Dicom Retrieval

Pop in the CD and run (mpi2d stg=1.1). Navigate to the head CD folder and hit ok.
This stage will find all files recursivly and sort them based on their header into injections and slices. Based on these slices it confirms that all the series numbers are sequential for that injection. If they're not(all slices for a given injection have the same series number) it will find new series numbers of the ones avaliable and rewrite them with that series number.

Pre-processing

Once the Reconstruction has been done make a soft link to the ReconData folder (ln -s Path/To/Recon/Folder/ReconData ReconData) then run (mpi2d stg=1.2)
This stage will open folders with more than 38 dicoms and create a par file for them, sorting based on the timestamp the image was marked with. It will upsample them, then find the RV and LV to know where to crop the heart out. Each slice gets it's own rangex, rangey. After it has processed all folders it retroactivly looks at the LV/RV location of the previous slices to find outliers and recrops them.

Batch processing

Find the script that loops through all par files(it will delete mpi2d.par before it searches(dir)) and do registration, then run mpi2d stg=3.11 for manual tweaking.
This script should run the model based segmentation(stg 3.12) to create preliminary contours.

Stages, their purposes and assumptions

1.1: Dicom Retrieval
This stage copies dicom files from a target folder recursivly and sorts them based on slice location and series description.
It assumes the present working directory is the patient folder, wherein it will create a DicomData folder and a Processing Folder. The former is where it places the sorted dicom files into folders with the following format: SeriesDescription(SeriesNumber).
This stage does not rely on mpi2d.par
Known Bugs: If the SeriesDescription and SliceLocation tags don't exist in the dicom header this function will fail. A hack is to rewrite dicom files copying the series number to the series description, and giving slice locations that increase with slices for an injection. This is also a good time to confirm timestamp information is present. A hack is to set the AcquisitionTime = AcquisitionNumber/2 or whatever function suits your fancy.

1.2: Par file creation, cropping limits, timestamp creation
This stage first checks for the existance of ../ReconData. If that folder does not exist it goes to ../DicomData and looks or folders which contain dicom files. For each folder that contains dicoms it will open each up, and extract timestamp information. It takes the seriesDescription tag from the header to look for other slices(folder names) with the same seriesDescription and derives from this the slice number. It then constructs par files. The rangex, and rangey, are croppings of the heart. This is done my giving the FindRVLV function the cinemri movie and centers a width/3 x height/3 window around the LV and stores this range in the par file. If a par file already exists the user will be prompted if he wants to re-use these values. Keep in mind that changing the rangex,rangey values will result in misalignment of pre-existing contours.
If you want to reprocess just a single dicom folder pass in the parameter into mpi2d stg=1.2 DoThisFolder='bla/blalkj'
This stage does not rely on mpi2d.par
Known Bugs: If the heart location is different for injections within this ReconData folder then there will be an incorrect cropping for the minority. Outliers are more than 1 standard deviation away from the mean position of the LV. Outliers are then cropped based on the median LV position.

0.23: Apply shifts
This stage assumes that for mpi2d.par there exists a shift vector. This stage will upsample the dicom images pointed to by the infile line in mpi2d.par and shift by the values in the shiftMan file, then crop based on rangex and rangey. It then saves out the cinemri1 file
Known Bugs: If the shift vector length is smaller than the total dicom files it will cause an error. This is indicitave of copying a shift vector from a different slice, or that the shift vector was created based on a cinmri file that had fewer frames than the number of dicoms. The shifting is done with a circshift(frame,shifts(t,:));

0.26: Force the shift vector to have [0 0] at referenceFrame
In many cases you want to have an arbitrary registration technique and some pre-existing contours. The contours are assumed to overlay the heart when there is no shifting required. This frame must be specified in the referenceFrame line in the parfile that corresponds to the shift vector you are trying to apply. In other words: You made contours based on a certian movie file. See which frame you didn't move anything. Write this number down. Do whatever registration algorithm you want and have it overwrite the shift file. Go to the par file and change the referenceFrame value to be what you wrote

down. This stage makes it so that whatever registration algorithm you want it will make sure the heart is where it needs to be so the contours cut it out
<u>Known Bugs:</u> This stage favors referenceFrame based registration algorithms. If you're using one that doesnt' have a referenceFrame and it messes up a little on the reference frame then the segmentation will be off

0.24: Copy shifts and contours
If you're doing batch processing many times you want to have the shifts you made in one slice apply to the other slices in the same injection, or the contours you made for a slice apply to all other injection's slices. This stage will copy the shift file to all other slices for that injection if they don't already have a shift vector. It will copy contours to all with the same slice number. This stage is a good template to copy from so you can sort all your date by injection or slice number
This stage does not rely on mpi2d.par
<u>Known Bugs:</u> no known

0.3: subset pre-Processing
This stage will copy files(shifts, contours, timestamp) from mother files to child subset in preperation for processing. This stage assumes that processing has been done for a patient and that a shift vector file and the contour files exist for a slice. Each subset will then get its own copy to simplify processing.
This stage assumes there is a Subset folder in the ReconData folder. Each slice has X number of children folders in this Subset folder, where X is the number of subsets that slice was split into.

```
ReconData
        Description1(2000)
                (DicomData)
        Subsets
                Description1(2001)
                        (DicomData created from subset reconstruction)
                Description1(2002)
                        (DicomData created from subset reconstruction)
                Description1(2003)
                        (DicomData created from subset reconstruction)
```

If you want to reprocess just a single dicom folder pass in the parameter into mpi2d stg=1.2 DoThisFolder='bla/blalkj'
This stage does not rely on mpi2d.par
<u>Known Bugs:</u> bugs

2.1: Complete Manual Registration
Obsolete. This stage can open up legacy files where the infile line in the par file points to mat files or raw files. A slash(/) indicates that it's a folder containing dicom files. It will upsample them, crop them, then if the rotNinety flag is set it will rotate them. This stage is highly likely to be incompatable with traditional processing.
It tries to do automatic registration with a buggy function then it saves out the shift vector.
<u>Known Bugs:</u> I have not tested the incompatability of this function, but this stage should not be run.

2.5: Mutual Information
This stage retrieves and upsamples images pointed to in the infile line of the par file(mpi2d.par), then performes mutual information registration. It then saves out the shift vector
<u>Known Bugs:</u> Not so much a bug, but a pain as this stage is extreemly slow. It also doesn't consider

shifts further out than 10 pixels, O(n^2*m^2) where M is the maximum shift considered)

2.6: Cross Correlation
This performes basic cross correlation to a reference frame specified in the par file
Known Bugs: This registration does well when there are only small contrast changes in the image. Thus it performes poorly during the transition between the RV recieving contrast agent and the LV recieving contrast agent.

2.8: Temporal Smoothing
This stage retrieves and upsamples images pointed to in the infile line of the par file(mpi2d.par), then performes temporal smoothing registration. It then saves out the shift vector.
This stage regesters frames to a temporally smooth version of the movie. It also does quasi-periodic smoothing and prediction of the shift vector it returns.
Known Bugs: This function relies on the timestamp information and will bug out if the timestamp has bad differences, or is transposed improperly.

2.9: Model Based Registration
This stage retrieves and upsamples images pointed to in the infile line of the par file(mpi2d.par), then performes temporal smoothing registration. It then saves out the shift vector.
This stage does model based registration and displays preliminary k-trans and delay information.
Known Bugs: no known bugs

3.01: Display contours overlaid on all slices and all injections
It first groups all par files into injections and slices. Then it retrieves the controus and cinemri movies for all slices and displays them into 1 figure with contours overlaid. Pause is used to advance to the next time.
Known Bugs: This stage shows all slices in 1 column. If you have only 1 injection and 8 slices there won't be enough vertical space to show them all. Change the subplot function to suit your needs(typically a = ceil(sqrt(nDatasets)) and b = ceil(nDatasets/a), and setting which subplot to some counter is sufficent.

3.1: Manual non-intuitive segmentation
This stage assumes the cinemri file is properly registered. You first pick a reference frame, upon which to base your contours. All steps are outlined on the output screen, but give no feedback as to what effect the contours you've made have on the curves.
Known Bugs: It creates contours properly, but there's no feedback or adjusting. You only have 1 shot and it better be good, or you'll have to recreate the reference frame, the AIF, the endocardium, the epicardium, the start angle all over again.

3.11: Tool to help review and tweak the contours and shift vector
This stage assumes the cinemri file exists. If the contour exist it loads them up. The user may then modify the contours and recieve feedback as to how those contours would extract tissue curves out and the AIF. You may modify the shift vector with the arrow keys. Upon quitting the user is asked if he wants to save the contours, which would overwrite the contours that exist. Also the user is asked if he wants to save the shifts. If you do know that it simply adds to the vector that already exists. If the cinemri1 files was created from another shift vector then the resulting shift vector after saving will not create the cinemri file that you just looked at.
Known Bugs: If the shift vector file is was used in a different manner than (frame = circshift(frame,shifts(t,:))) then this will break the existing shift vector. There is no consistancy check

to make sure that the shift vector specified in the shift vector results in the image you're seeing on the screen. It simply has circularly shifted the cinemri frame.

3.12: Auto-Segmentation of the endocardial, epicardial, and LV contour
This stage will attempt to do auto-segmentation and contour creation. This stage will create an eronious start angle if no RV segment exists this program cannot find a proper angle to the apex.
It will save the contours out and display the contours to figures.
Known Bugs: This function needs a better check to make sure there is an inside contour and an outside contour. It also discludes eschemic regions.

3.2: Apply the contours to the cinemri and extract the tissue curves
```
Myo = (outtermask - innermask)>0;
```
These masks are created from roiPoly(arbitrary frame, epi(:,1), epi(:,2)).
The center is the midpoint of the outermask or epicardium. The AIF curve is the 5th highest value in the AIF mask.
Known Bugs: Possible that this stage divides the regions up differently than the previous incarnation of tissue extraction stages(mpi_applyRegions.m)

3.22: Blind AIF estimation
Assumes the contours and cinemri file is regiesterd and exist. This stage will replace the AIF line of the deltaSIcurves file(first row) with an estimation of the AIF based purely on the tissue curves cut out by the segmentation. The resulting deltaSIcurves file has a series number that's the same as the input, but shoud be modified to indicate the difference
Known Bugs: no known bugs. It overwrites the deltaSIcurves file that already exists.

3.21: Review of the contours and cinemri file
Assumes the cinemri file exists and is regestered as well as the contours. It will display the contours overlaid on the cinemri movie and pressing keys lets you progress through the movie to see if the registration was good as well as the contours.
Known Bugs: no known.

3.3: Display the Tissue curves
Pretty simple. Display in figure 1 the tissue curves
Known Bugs: bugs

3.4: Convert tissue curves to normalized deltaSIcurves
According to the numtissue variable it will average that number of tissue values and subtract that from the entire curve so that they represent the difference from an intial frame.
Known Bugs: none

3.41: Convert to deltaSIcurves witafter Proton Density correction has been applied
Assumes the proton density files are there. I don't know what this implies.
Known Bugs: -

4.1: Fit the deltaSIcurves with the 2-compartment model
Assumes the deltaSIcurves file is there, as well as the timestamp file for interpolation. Saves out a flow file(text) that has the values of the model parameters. Different files are saved out based on how the deltaSIcurves file was created(it's own AIF, subset analysis, how much it scaled the AIF...)
Known Bugs: no known

4.2: Display the 2-compartment model's parametes on a figure with myocardial mask
Assumes the fits have already been computed. It has a black background and each of the azimuthal regions is colored based on the k-trans value it has
Known Bugs: no known

4.3: Compute Upslope model
Based on the deltaSIcurves file(pixelwise or regional) it computes the ratio of the maximum upslope of each tissue curve over the maximum upslope of the AIF. This slope is calculated based on a 5 frame window with linear regression. It saves out an upslope file.
Known Bugs: no known bugs. Due to linear regression it is sensitive to outliers.

4.31: Display the upslope file
Based on the upslope file(pixelwise or regional) it displays the ratio of the maximum upslope of the tissue curve over the maximum upslope of the AIF. If pixelwise was done then each pixel is colored according to the upslope ratio. Othwise the entire region is colored accordingly.
Known Bugs: no know bugs

5.1: Compute Fermi model based on the deltaSI curves
I'm not versed in the functionality of this stage, nor its flaws.
Known Bugs: Unknown

5.2: Display the Fermi model parameters
Never tried this stage
Known Bugs: Unknown

6.1: Display k-trans with variable width transparency over the cinemri's referenceFrame
The last 2 plot lines draw the contours in a smooth fasion(smoothed with a gaussian).
This stage assumes a k-trans flow file, cinemri file, and contours exist.
There are 3 tweakable parameters for this stage: Alpha, HorzOffset, and MyoFatness.
Alpha: Adjusts the maximum opaqueness of the overlaid k-trans color within the myocardium
MyoFatness: Adjusts how wide the k-trans overlay is within the myocardium. The default is 1
HorzOffset: Adjusts how much to offset the k-trans text overlay. Sometimes the text ends up off the chart. This helps bring it back.
Known Bugs: no known bugs

0.xx: Template Stage
stuff about that stage
Known Bugs: bugs

Useful files

fixshifts.m
This file takes the shift vector as it has been created by an automatic method, or manual if you like, and the timestamps, and tries to find a set of frequencies that the breathing is happening at. It then replaces the outliers of the shift vectors with this prediction. It rotates the shift vector to a line, assumming the motion is mostly 1 directional, and does an fft on the signal from an interpolation based on the timestamp. After finding the primary 4-5 frequencies it inverses it to the real world and then it finds the difference and fixes the shifts. This is really helpful if your algorithm explodes ever so often.

FindLVRV.m
This function simply takes in the cinemri movie and returns the position of the RV and the LV. It does this by scoring a set of candidate points(regional maximums of temporal maximum) and picking the points with the highest scores. Then it does k-means on the delays to distinguish the LV from the RV. It returns the median point in each cluster.
This function is helpful if you need to know a point in the RV for model based registration or simply need the LV when you don't have the contours at hand.

model.m
This function takes in a the AIF function that can be used to predict the rest of the pixels(typically the RV), a time delay window upon which to assume all delays to be contained, the cinemri, and the mask of which pixels to model.
It then finds the deltaSIcurves, does a first order approximation to solving the equation of the 2-compartment model and returns a plethora of useful stats.
parValues help with segmentation, finding less-healthy regions, whatever, but are typically not useful in getting a true approximation to the true k-trans values for those tissues.
delays are a pixelwise image of the delays for each pixel when the onset of the curve happened(maximum upslope) and will be within the TimeDelayWindow.
smoothed: this is a smoothed version of the cinemri movie. Helpful in registration and viewing.

gatherFilesRecursivly.m
This function will start from the present working directory and gather files recursivly and return a cell array(1d) of strings of the path from the present working directory down to the each file it can get access to. Helpful in finding dicoms.

updateParFile.m
This file is helpful if you want to programatically make a small change to a par file. Create a struct and set a field to a string version of whatever value you want to change. Then pass in the filename of the par file and the struct

```
myparfile = 'series45.slice2.par';
clear temp
temp.rangex = '25:150';
updateParFile(myparfile,temp);
```

ReadPar.m
This is a special script. It will populate the workspace with variables in the par file who's name is specified by the variable ParFileName. For example

```
ParFileName = 'series59.slice1.par'
```

```
ReadPar;
disp(infile);    %this line results in matlab displaying the path to the dicoms specified in the par file.
It is useful when you want to get at the slice number of whatever par file you're looking at without
needing to parse the filename.
```

edscriptP0505.m

This script has been streamlined to facilitate creating subsets and processing them. Look at the first few variables set to see which ones need to be modified. Each hard coded section of variables starts with seriesnumStart and ends with MID. Modify each of these as needs be. Cross check the series description of the injection you're doing, write down the seriesNumber for these and see which MIDs they match up with by looking at the RawData (.dat files).

AIFViewer.m and AIFViewerSubsets.m as well as the multiSRT versions

These files help in reviewing which slice number has the highest AIF. The variable injectioratio must be changed so that the AIFs will be scaled properly.