Information Processing for Medical Imaging
MPHY0025 – 2022/2023

Registration exercises 2

Python version

You will need the same python libraries installed in your environment as for exercises 1, i.e.:

scikit-image

matplotlib

scipy

numpy

You have been provided with some utility functions in *utils2.py* for use in these exercises as well as a template script for your solutions. Some of the functions in *utils2.py* are the same as in *utils1.py*, but there have also been some updates to some of the functions (so that they use `numpy.array` instead of `numpy.matrix`), and there are also some new functions. You should look at the code for the utility functions and make sure you understand what they do and how they work.

You have also been provided with the following 2D images:

ct_slice_int8.png – an axial CT slice of a head, stored as unsigned 8-bit integers

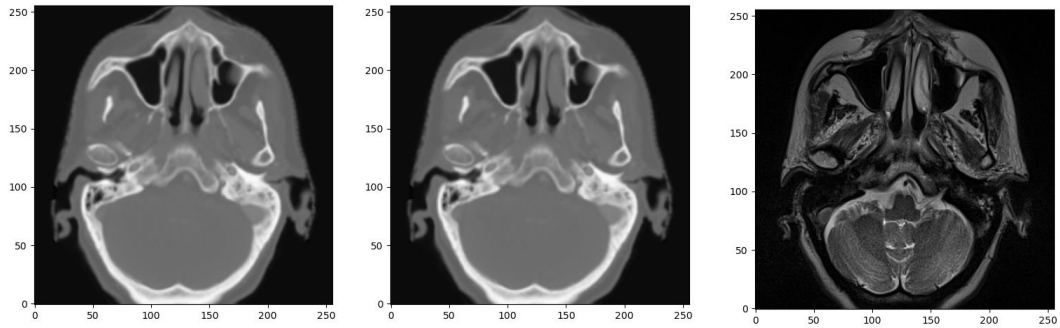
ct_slice_int16.png – the same axial slice CT slice, stored as unsigned 16-bit integers

mr_slice_int16.png – the corresponding axial MR slice, stored as unsigned 16-bit integers

For the animations in these exercises to display correctly in python the figures need to be displayed in separate windows and not in the console (or notebook if using notebooks). This can be achieved by setting the *matplotlib* backend to *qt*, as done at the top of the template script. Note – in exercises 1 we set the backend to *auto* but I discovered this does not display the animations correctly on Mac. Setting the backend to *qt* seems to work on all platforms (that we have tested so far!).

Similarity measures

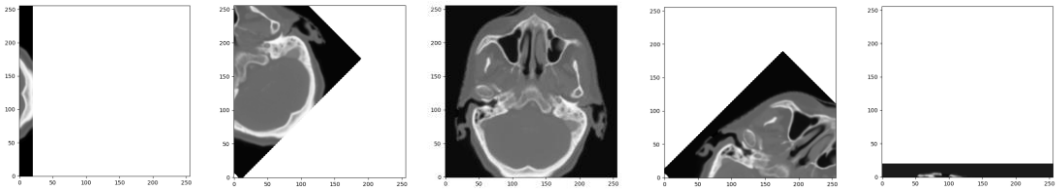
The template script includes code to load in the three 2D images. Add code to the template script to display the data type of each image and check these match the expected data types. Now add code to the template script to convert the images to double, reorientate them into 'standard orientation', and display each image in a separate figure using the `dispImage` function from *utils2.py*. The images should appear like this:



You should notice that the two CT images appear exactly the same, but if you examine the actual values in the images you will see that the images use different intensity ranges. If you move your cursor over the image the intensity values can be seen at the top right of the figure:

In the 16 bit version, the intensity values correspond to Hounsfield Units (or rather Hounsfield Units + 1000, since air has a value of -1000, but the image cannot contain negative values as it is stored as unsigned integers), but in the 8 bit image the values have been scaled as the image can only contain values from 0 to 255.

The template script contains some code to rotate each of the images between -90 degrees and 90 degrees, in steps of 1 degree. Edit the code so that on each iteration of the loop it uses the `affineMatrixFromRotationAboutPoint` function from *utils2.py* to create an affine matrix representing an anti-clockwise rotation by theta degrees about the point 10,10, and uses the `defFieldFromAffineMatrix` function to create the corresponding deformation field. Then resample each of the 3 images using the `resampImageWithDefField` function and display the transformed 8-bit CT image using the `dispImage` function. If this has been implemented correctly the image should appear to rotate clockwise, starting with a rotation of -90 degrees (so mostly being 'off to the left' of the original image) and finishing with a rotation of +90 degrees (so mostly being 'off the bottom' of the original image), as shown in the intermediate images below:



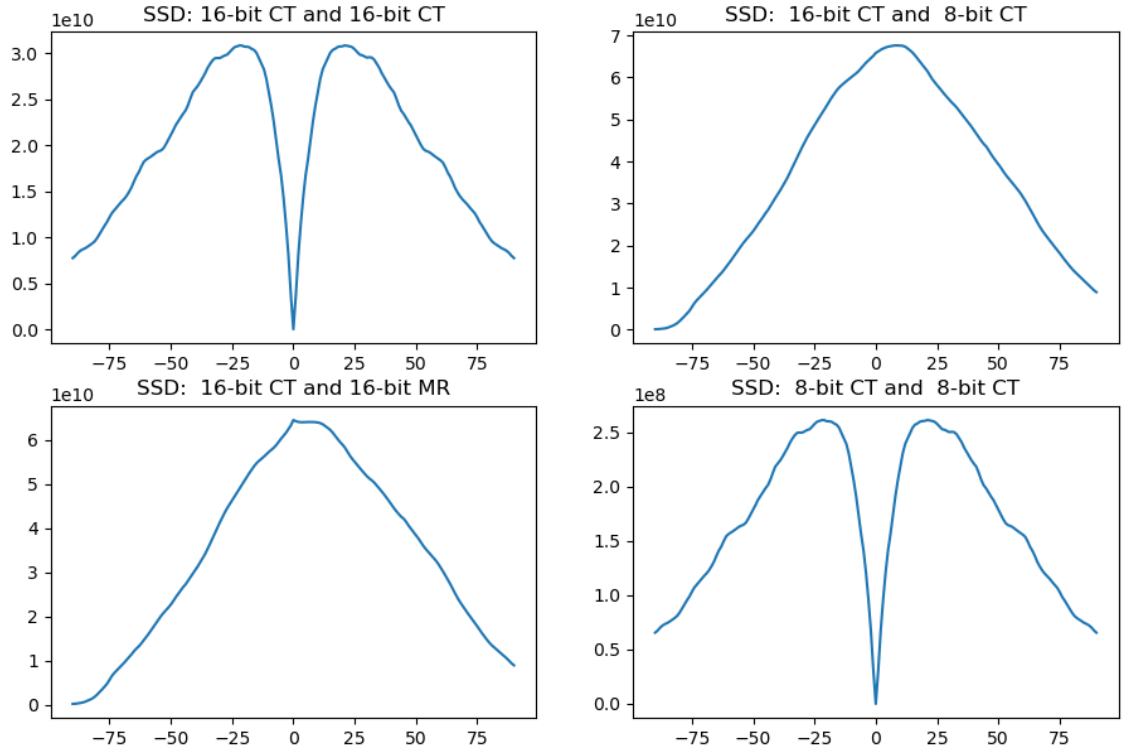
Make sure you understand why the image appears to rotate clockwise when the function produces an affine matrix representing an anti-clockwise rotation.

Note, the default padding value of NaN should be used when resampling the image so that pixels from outside the original images are ignored when calculating the similarity measures

below. Now modify the template script so that on each iteration of the loop it calculates and stores the SSD between:

1) The original 16-bit CT image and the transformed 16-bit CT image
2) The original 16-bit CT image and the transformed 8-bit CT image
3) The original 16-bit CT image and the transformed MR image
4) The original 8-bit CT image and the transformed 8-bit CT image

Now rerun the cell with the for loop (you may want to comment out the lines that display the image and pause so that the code runs faster). Once you have done this run the next cell in the template script that plots the SSD values on the y-axis against the angle theta on the x-axis, for each of the four cases above. The plots should look like:



Note that:

SSD reaches a minimum (of 0) for cases 1 and 4 when the images are in alignment.

For cases 2 and 3 SSD does not have a minimum when the images are aligned.

For all cases the SSD decreases as the overlap between the images decreases.

Although the shape of the SSD curve for cases 1 and 4 is the same, the values of the SSD are different by 2 orders of magnitude.

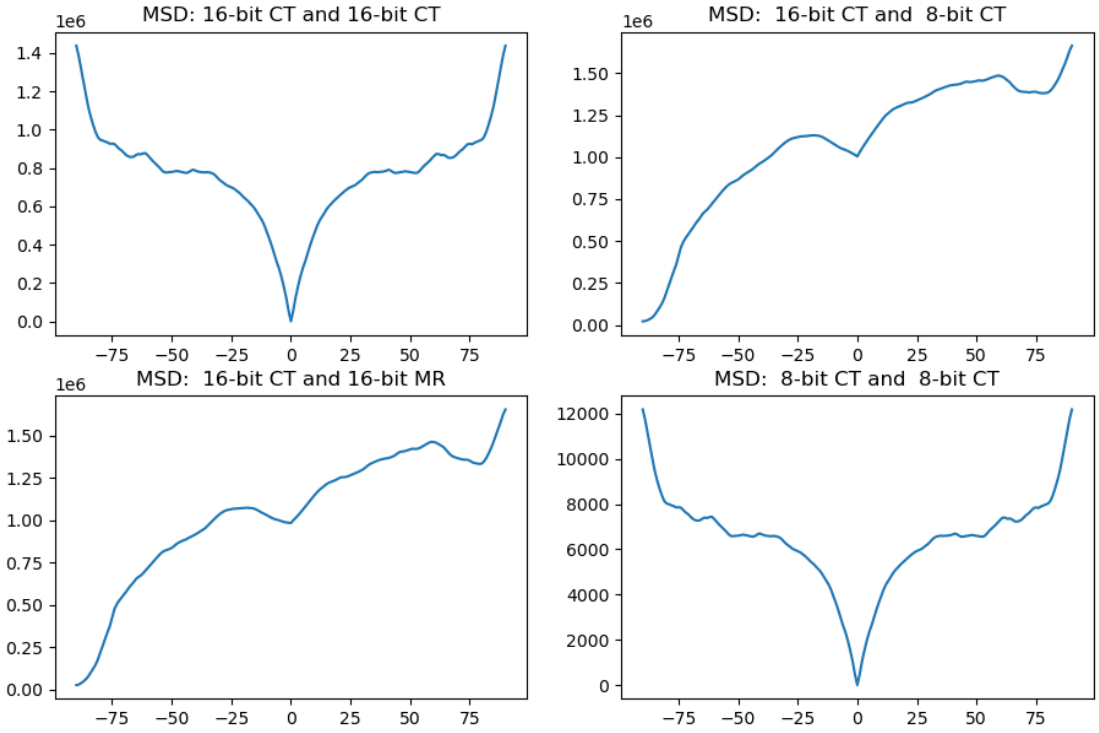
Make sure you understand why you get these results. If you are not sure ask me or one of the PGTAs in the lab session.

Now edit the code in the template script so that it rotates the images as above but calculates the MSD instead of the SSD at each iteration, and then plots the MSD values. The plots should look like those on the next page. Make sure you understand why:

The MSD for cases 1 and 4 does not decrease as the amount of overlap decreases.

The shape of the MSD curves for cases 1 and 4 are the same but the values are larger for case 1.

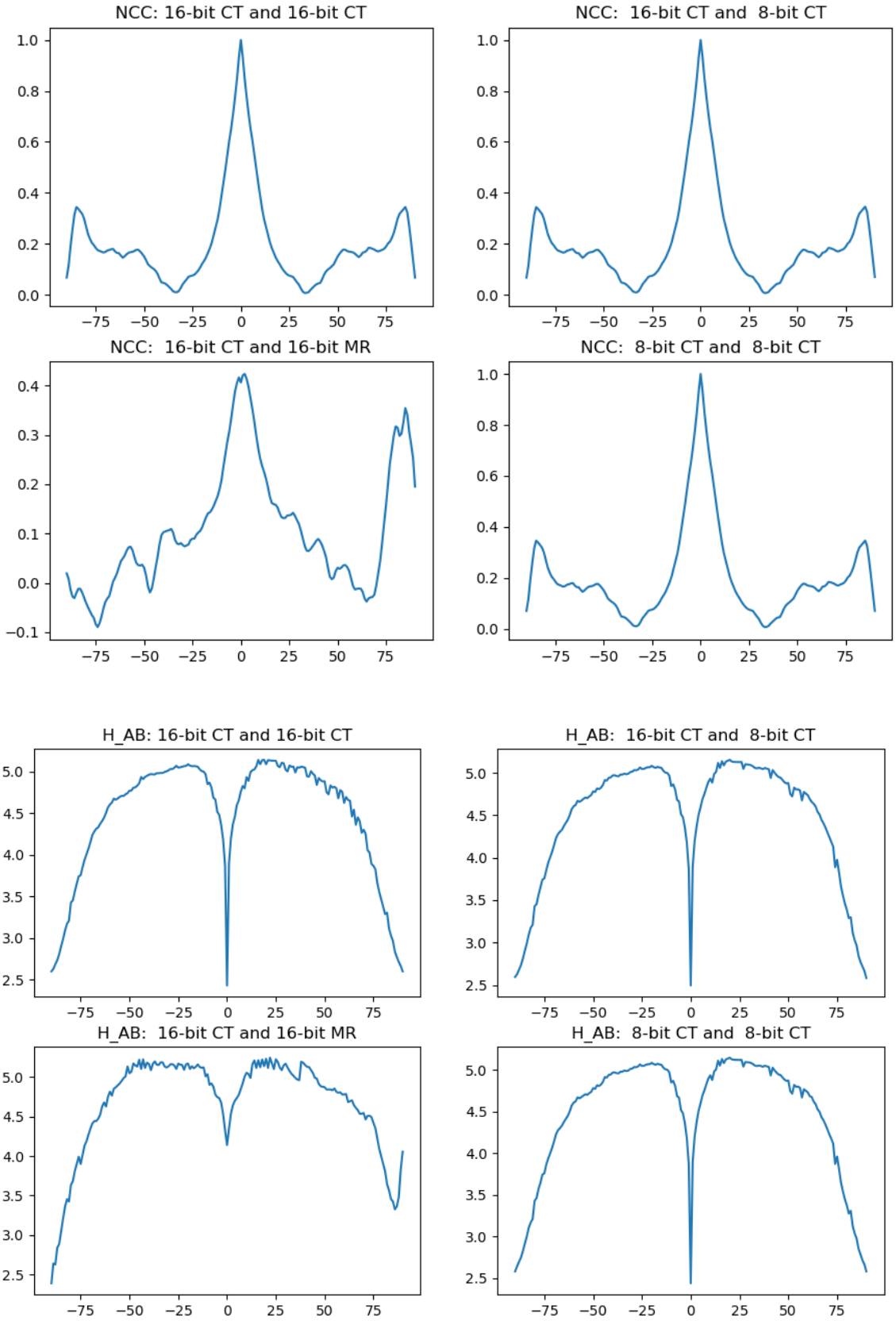
The MSD values for cases 2 and 3 are lower for negative values of theta and higher for positive values (this one is a bit tricky!).

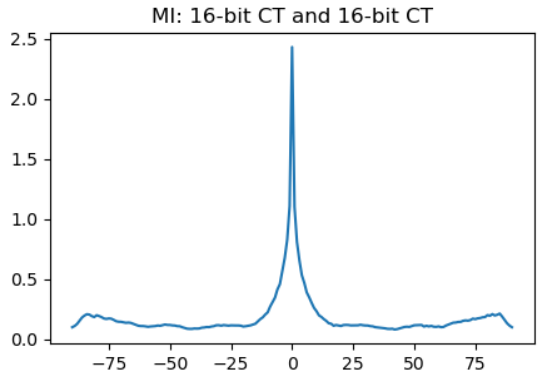


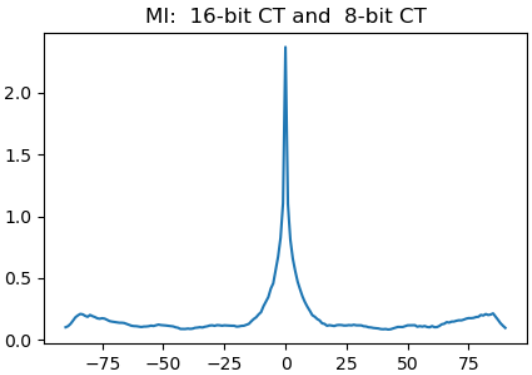
Now edit the template script to implement the `calcNCC` function so that it calculates the Normalised Cross Correlation (NCC) between two images. Then edit the template script so that it uses your `calcNCC` function and the `calcEntropies` function from *utils2.py* to calculate the (NCC) and the joint and marginal entropies (H_AB, H_A, and H_B) instead of the MSD/SSD at each iteration. Add code to the template script to calculate the Mutual Information (MI) and Normalised Mutual Information (NMI) from the entropy values, and plot the results for NCC, H_AB, MI, and NMI. The plots you get should look like those on the following pages.

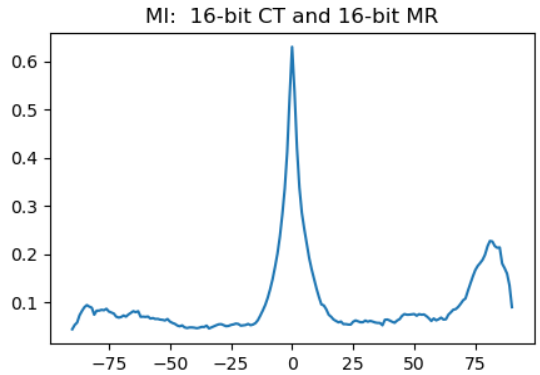
Do the different measures perform as expected for the different cases? Based on these results, which measures are suitable for registering the different pairs of images? Does this agree with what you were taught in the lecture?

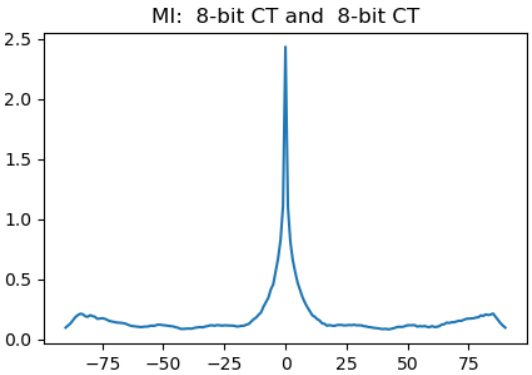
Make sure you understand all the results you get.

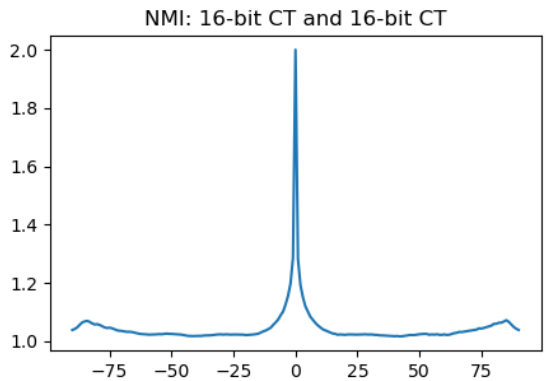

NCC: 16-bit CT and 16-bit CT
NCC: 16-bit CT and 8-bit CT
NCC: 16-bit CT and 16-bit MR
NCC: 8-bit CT and 8-bit CT
H_AB: 16-bit CT and 16-bit CT
H_AB: 16-bit CT and 8-bit CT
H_AB: 16-bit CT and 16-bit MR
H_AB: 8-bit CT and 8-bit CT

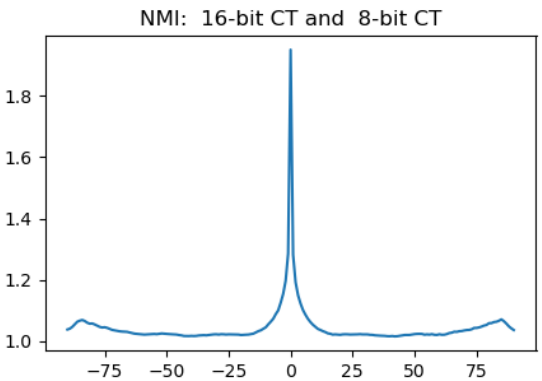

MI: 16-bit CT and 16-bit CT
2.5
2.0
1.5
1.0
0.5
0.0
−75 −50 −25 0 25 50 75

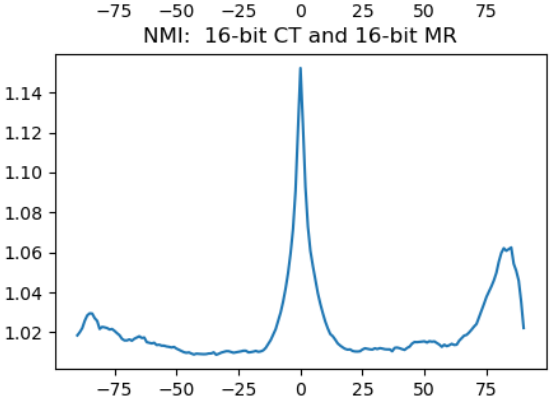

MI: 16-bit CT and 8-bit CT
2.0
1.5
1.0
0.5
0.0
−75 −50 −25 0 25 50 75

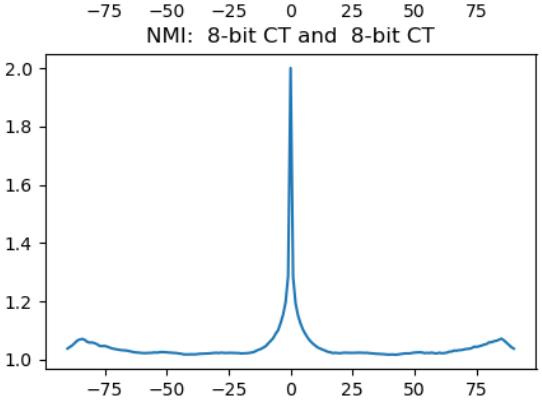

MI: 16-bit CT and 16-bit MR
0.6
0.5
0.4
0.3
0.2
0.1
−75 −50 −25 0 25 50 75


MI: 8-bit CT and 8-bit CT
2.5
2.0
1.5
1.0
0.5
0.0
−75 −50 −25 0 25 50 75


NMI: 16-bit CT and 16-bit CT
2.0
1.8
1.6
1.4
1.2
1.0
−75 −50 −25 0 25 50 75


NMI: 16-bit CT and 8-bit CT
1.8
1.6
1.4
1.2
1.0
−75 −50 −25 0 25 50 75


NMI: 16-bit CT and 16-bit MR
1.14
1.12
1.10
1.08
1.06
1.04
1.02
−75 −50 −25 0 25 50 75


NMI: 8-bit CT and 8-bit CT
2.0
1.8
1.6
1.4
1.2
1.0
−75 −50 −25 0 25 50 75