Distortion Detection Script

User Guide and Technical Manual

User Guide

1. Introduction

This script makes use of the HPD phantom in order to automatically detect the fiducial spheres within and compute distortion. The phantom consists of 5 plates each with each plate containing 4, 13, 21, 13 and 5 spheres each. Each sphere is 40mm apart this includes both in between and within plates. The phantom has many other features such as T1 spheres, in this script only the fiducial spheres (light blue) are considered. Each plate with the fiducial spheres is shown below, the spheres highlighted in white. This code will find the position of each sphere and compute distances both inter (between plates) and intra (within a single plate). By comparing the measured distances with the known, several distortion metrics can be computed and displayed.

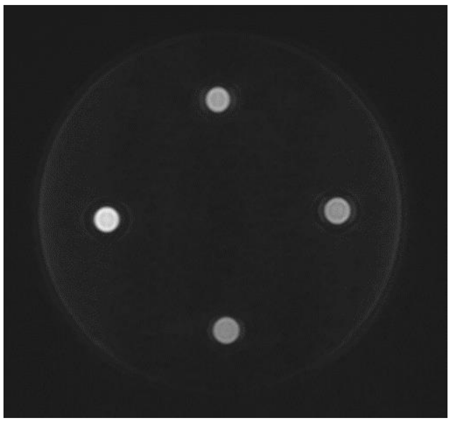
Figure 1: Plate 1 with 4 spheres

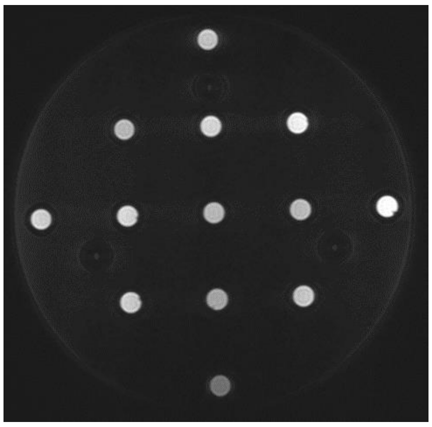
Figure 1: Plate 2 with 13 spheres

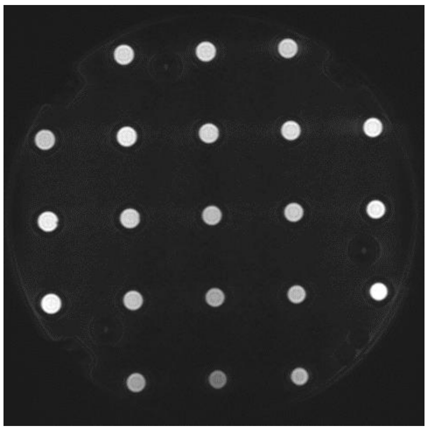
Figure 3: Plate 3 with 21 spheres

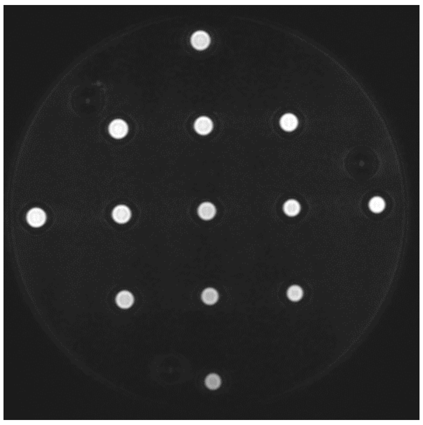
Figure 4: Plate 4with 13 spheres

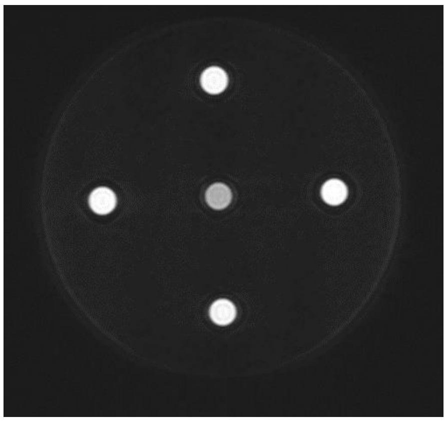
Figure 5: Plate 5 with 5 spheres

2. Requirements

The following requirements are needed in order to run this python code.

- Python 3
- Numpy
- Scipy
- Matplotlib

- Pandas
- Glob
- Pydicom
- Skimage
- Sklearn

The easiest way is to try running an example, if anything is missing use pip install <package> to install the missing packages.

3. Code Acquisition

The latest code can be obtained at https://github.com/DrSpacemanMD/Distortion-Detection. It can be directly downloaded from https://github.com/DrSpacemanMD/Distortion-Detection/archive/refs/heads/main.zip or the git repository can be checked out using the `git clone https://github.com/DrSpacemanMD/Distortion-Detection.git` command.

4. Phantom Set-up

It is important that the phantom is set up as discussed here to ensure the script can successfully detect the spheres. Firstly, ensure the phantom has sufficient water, if a sphere is not enclosed in water this may cause issues with detection. Place the phantom in the scanner ensuring the side with the 4 spheres (figure 4) is pointing towards the ceiling. Using the lasers align the spheres so they are flat and level with as little rotation as possible. Using the head coil scan, the phantom. It is important to ensure the phantom is in the centre of the field of view. For optimal results ensure geometry correction is on. On conclusion of the scan, note the sequence of interest and push the DICOM files to your device. Save all the DICOM in a folder located in same location as the distortion scripts.

5. Quick Start Guide

In order to compute distortion a series of python functions must be called. This is for all intents and purposes considered as an input script. An input script contains two main components:

- The DistortionCalculation class.
    - This class computes the location and distances between each sphere.
- The AnalysisResults class.
    - Takes the previously computed distances and calculates a variety of metrics for analysis.

There is a wide variety of dependencies required, majority will be included within Python. The easiest way to install

The first step is to include the appropriate modules as shown below.

```python
import Analysis
import Compute_Distortion
```

The next step is to load the DistortionCalculation module, passing in the folder name with the DICOM and the sequence of interest. Be sure to change the arguments to the correct folder and sequence name.

```
ComputeDistortionGeoCorrection =
Compute_Distortion.DistortionCalculation("FolderName", "SequenceName")
```

In a similar fashion the analysis module must be set up as the follows, passing the analysis name (used to name output files etc) and the previously initialised DistortionCalculation module.

```
NoDistortionAnalysis =
Analysis.AnalysisResults("AnalysisName",ComputeDistortionGeoCorrection)
```

Now that we are set up call GetFudicialSpheres to find the position of all fiducial spheres in the phantom.

```
ComputeDistortionGeoCorrection.GetFudicalSpheres()
```

Now the spheres are located we can find the distances between them, both inter and intraplate.

```
ComputeDistortionGeoCorrection.GetDistances()
```

At this stage all distances have been computed so next we want to perform some analysis. So, call the analysis modules DistortionAnalysis function. This computes all the metrics which we can use to determine how much distortion exists.

```
NoDistortionAnalysis.DistortionAnalysis()
```

Finally, to see what the values of the metrics are call the PrintToScreen method

```
NoDistortionAnalysis.PrintToScreen()
```

At this stage a wide variety of metrics print to screen. To run this script through your chosen python interpreter, often typing python <file name> is sufficient. For a more complete description of each function along with others not discussed here please see the function reference section. The complete quick start script is shown below. This script is ready to go with the TestData included in the github.

```
import Analysis
import Compute_Distortion
import sys

#Set up the distortion calc by passing the folder with all the DICOMS and the
sequence of interest.
ComputeDistortionGeoCorrection =
Compute_Distortion.DistortionCalculation("TestData" , "3D Sag T1 BRAVO Geom Core")

#Set up the analysis script, this takes the computed distacnes and output metrics.
Pass a tag for the calculaton (used for naming saved images etc) and the previously
constructed distirtion calc class
NoDistortionAnalysis =
Analysis.AnalysisResults("No_Distortion",ComputeDistortionGeoCorrection)

#Call this function to get all the centre of the distances
ComputeDistortionGeoCorrection.GetFudicalSpheres()


#Call this function to calculate the distances
ComputeDistortionGeoCorrection.GetDistances()

#Call this functuion to analyse the distances and compute the metrics
```

```
NoDistortionAnalysis.DistortionAnalysis()

#Call this function to print the computed metrics to screen
NoDistortionAnalysis.PrintToScreen()
```

6. Metrics Computed

Within is a list of metrics computed by the script. Please note X is in the Sagittal direction, Y the Axial and Z the Coronal.

- Interplate Max Distortion
  - The largest interplate difference between the measured and expected distance, computed in mm.
- Interplate Max Percentage Distortion
  - The largest interplate difference between the measured and expected distance, computed in percentage.
- Interplate Coefficient of Variation
  - The Coefficient of Variation is computed for all interplate distances
- Interplate Max Distortion X
  - The largest interplate difference in the x direction between the measured and expected distance, computed in mm.
- Interplate Max Distortion Y
  - The largest interplate difference in the y direction between the measured and expected distance, computed in mm.
- Interplate Max Distortion Z
  - The largest interplate difference in the z direction between the measured and expected distance, computed in mm.
- Interplate Coefficient of Variation X
  - The Coefficient of Variation in the x direction is computed for all interplate distances
- Interplate Coefficient of Variation Y
  - The Coefficient of Variation in the y direction is computed for all interplate distances
- Interplate Coefficient of Variation Z
  - The Coefficient of Variation in the z direction is computed for all interplate distances

- Intraplate Max Distortion
  - The largest Intraplate difference between the measured and expected distance, computed in mm.
- Intraplate Max Percentage Distortion
  - The largest Intraplate difference between the measured and expected distance is computed in percentage.
- Intraplate Coefficient of Variation
  - The Coefficient of Variation is computed for all Intraplate distances
- Intraplate Max Distortion X
  - The largest intraplate difference in the x direction between the measured and expected distance, computed in mm.
- Intraplate Max Distortion Y
  - The largest intraplate difference in the y direction between the measured and expected distance, computed in mm.
- Intraplate Max Distortion Z

    - The largest Intraplate difference in the z direction between the measured and expected distance, computed in mm.
- Intraplate Coefficient of Variation X
    - The Coefficient of Variation in the x direction is computed for intraplate distances
- Intraplate Coefficient of Variation Y
    - The Coefficient of Variation in the y direction is computed for intraplate distances

7. Function Reference Guide

Within this section each public function is listed with a description of the functionality. For detailed working please consult the technical manual section.

1. Compute_Distortion functions
    a) GetFudicalSpheres()

        Function detects and computes the position of all fiducial spheres within the system. Must be run before any distances or analysis is computed. The search distance (see technical manual) can be adjusted by changing the searchWidth variable. Please note this distance is given in mm. This method makes use of binary images, this can be constructed 4 different ways. The method can be adjusted by setting the BinariseMethod variable to the correct name (as shown next). The methods of computing the threshold to binarise an image within is:

        1. Min
           Uses the Scipy threshold_minimum, this method is useful and fast but has been known to fail. If this happens try one of the other methods
        2. Constant
           Set a constant value to threshold the images, to do this set the Threshold variable.
        3. RatioOfMax
           This method considers the maximum value in each image then the threshold is set at a fraction of this. The fraction is set by the user in the ratio variable.
        4. IslandChecker
           This method iterates over several thresholds checking for instances when the correct number of objects (spheres) are found in the image. Based on this a threshold for each image is chosen. This method is more sensitive to background noise so often a tighter search window is suggested (set the searchWidth, default of 4.688)

    b) GetDistances()

        Takes the previous computed sphere centres and find the inter and intraplate distances.

    c) AdjustPoint(PointGuess,NewPoint)

        Allows the user to adjust any of the detected points should they be mis detected. The user inputs a point guess, the coordinates of the point can be found by using the CheckPoints function, the format is a list of [x,y,slice]. The user also supplies a new point from which the point that is closest to the PointGuess value then has its coordinates reassigned to the NewPoint variable.

    d) GetSagSlice(SliceNumber)

        Return a pixel matrix for the Sagittal slice at the given slice number

    e) GetAxialSlice(SliceNumber)

        Return a pixel matrix for the Axial slice at the given slice number

    f) GetCorSlice(SliceNumber)

Return a pixel matrix for the Coronal slice at the given slice number

2. Analysis functions
   a) DistortionAnalysis
      Computes all distortion metrics (outlaid in section 6) based on the distance computed previously. When calling this an image of each plate showing some key intraplate distances and one image showing key interpolate distances are also produced and saved.
   b) PrintToScreen()
      Outputs all the metrics to screen.
   c) CheckPoints()
      Allows users to check if the points detected are reasonable. When this function is called a window opens that contains a Coronal slice at position 0. By using the right and left keyboard buttons the slices can be increased or decreased. Points are shown on the closest slice as blue crosses. By hovering over a cross the x,y coordinates can be determined and the slice number determined by the title of the image.
   d) Show3dModel(metric=None)
      If no argument is used this displays a 3d plot of all the sphere positions, each colour represents a different plate. If a metric is used as an argument (for example interpolate max distortion, see section 6 for all metrics) then the two points that corresponds to that measurement is shown. If an intraplate metric is used it shows the connecting points in each plate. Useful for identifying the general plane which the distortion occurs in.
   e) OutputPeriodicData(file)
      Function designed to maintain a record of distortion data over time. If the file (as given in the argument) does not exist, the function makes a new file and adds the computed metrics to the file in a comma separated format. If the file does exist, the current metrics are appended. Each entry has a study date time and can be plotted externally or by using the plot CSV function.
   f) OutputToFile(filename)
      Output the metrics to a file, format is identical to PrintToScreen.
   g) PlotCSV(file)
      Plots the CSV file produced by OutputPeriodicData. Will save the plots in the “Plots” folder each has a prefix which is identical to the analysis name set in the constructer.
   h) GetMetrics()
      Return a list of all metrics computed.

8. Bug and Issue Reporting

If any bugs or issues are found, then it is suggested a new issue is made on the github page (https://github.com/DrSpacemanMD/Distortion-Detection/issues) with as much information about the bug as possible being entered. If any data is required, then a link to the data would also be useful.

9. Potential Solutions to issues

a) The sphere centres are wrong.
   Use the CheckPoints and AdjustPoints function to adjust any points.
b) Can I plot the csv elsewhere?
   The csv is standard format and can be easily loaded into excel.
c) Some points are missed, and the phantom is somewhat rotated.
   Try increasing the DistortionCalculation.searchWidth variable to increase the search width (see the technical manual)
d) My phantom was upside down or badly rotated how can I fix that?
   Not easily, in principle you could adjust the code to expect an upside-down phantom but it may be easier to just re-run the scan. If your phantom is heavily rotated then it is quite difficult to identify only one set of spheres, its probably best to re-run the scan.
e) I put all my DICOMs in a folder and it is taking ages to run, why?
   The code looks through each DICOM file and extracts the ones which are using the sequence defined. You can speed this up by extracting the relevant DICOMS and putting them in their own folder.
f) It has detected some weird points or missed them.
   Try another BinariseMethod and play with the associated setting.

10. Example Files

A series of example files are made showcasing some of the features.

a) QuickStart.py
   Quick start as shown in section 5.
b) Example_Check_And_AdjustPoints.py
   Demo of how to check and adjust any points if they are miscalculated.
c) Example_Plot3d_Metric.py
   Demo of 3d plotting showing how to plot a metric on the plot.
d) Example_Build_and_Plot_Historic_data.py
   Demos how to plot time series data where a csv is updated each time a new test is run.
e) Example_ShowSlices.py
   Demos how to take slices out of the 3d volume.
f) Example_BinaryMethods.py
   Demo of the various binary methods that can be used.

Technical Manual

This section discusses the actual underlying principles when detecting and computing distances between spheres. The computation of each metric is also discussed. Each DICOM with the defined sequence is loaded into a 3d volume from which any slice can be extracted. This volume is then used to detect spheres and hence compute distances.

1. Sphere Detection

We assume the phantom is centred in the field of view; from this we can guess where each plate lies. Since we know that each plate is 40mm apart we can guess where each plate lies from the centre. For a given plate we define a search distance which is centred around the guess we just worked out. For each image within the search distance a threshold which produces a binary image is found. This produces a series of thresholds throughout the search width. The exact method depends on the binarise method (see section 7.1.a in the user guide). Either a single threshold is used (based on the lowest detected threshold) or in the case of IslandChecker a separate threshold is used for each slice in the search width.

Then we reiterate through the search volume and binarise each slice using the threshold just obtained. From this we can extract all points which were above the threshold producing a list of 3d points which are enclosed within the sphere. Since this is a list, it is unclear what point is associated with what sphere. Since we know how many spheres should exist on this slice, we can use k-means clustering to associate each point with a sphere. Now that we know what points belong to what sphere we can simply compute the centre of mass for each sphere and hence the centre. This is repeated for each plate producing a list where each element contains the centre of each sphere in that plate.

2. Computing Distances

Distances are grouped into interplate (in between plates) or intraplate (within a single plate). The method is broadly similar but has a few differences.

a. Intraplate Distances
For each plate all the coordinates for each sphere are allocated into a list. We make use of a grid system. The grid size depends on the plate, and example of the grid is shown in figure 6

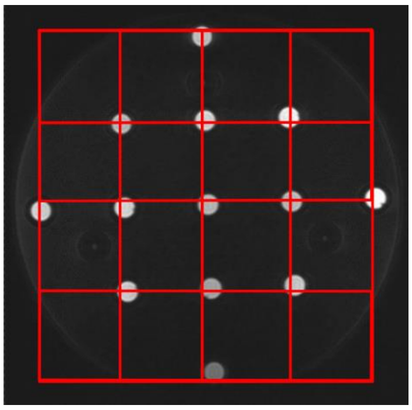

Figure 6: Plate 2 with the 5x5 grid

Using k-means clustering each point is assigned a grid point. We then iterate over each point from left to right and compute distances to all other points in the plate. Please note the distances are computed in 3d even though we only consider one plate at a time. Based on the number of grid points between the two points we can then compute the expected distance. The grid points, coordinates, measured and expected distance are stored for later use. Any duplicate distances (eg 1-2 and 2-1) are removed and all distances are stored for later analysis. In the outputted images only a small selection of sensible distances is shown but, in the metrics, outputted all distances are considered (including the long distances across the plate).

b. Inter plate Distances

The interplate distances are computed in a similar way, in this instance the grid system is extended to 3d with each sphere being associated with a 3d grid point by once again using k-means clustering. As with intraplate we go through each point and compute the distances to all other points. In these only distances that are in a different plate are considered. In the image saved the average distance between spheres of adjacent plates are drawn. In the metric output calculation, all distances are considered.

3. Metric Calculations

A variety of metrics is computed to assist with analysis. These are outlined in section of the user guide.

a. Max Distortion

The max distortion is computed by iterating over all distances computed and calculating the absolute difference between the measured and expected distance. The maximum deviation in mm from the expected distance is then computed for this metric. This metric is computed in both intra and interplate in an identical way. Although in the case of intraplate the maximum deviation over all plates is reported. Additionally, this metric is computed using the x, y and z components from each point in a similar way. This produces a max deviation in each axis, again repeated for intra and interplate distances.

b. Max Percentage Distortion

This is computed in a similar way to max distortion except the distance is normalised to the expected distance and multiplied by 100. This produces a percentage difference as opposed to an absolute difference in mm. In a similar way to max distortion this is computed for both inter and intraplate as well as in each axis direction (x,y and z).

c. Coefficient of Variation

The coefficient of variation is computed for distances which are exactly 40mm apart from each other. This is computed by considering the ratio of the standard deviation and the mean. This is conducted for both inter and intra plate. In the case of interplate x,y and z is computed but in the case of intraplate only x and y are considered since the variation in z is 0.