PhillipsPhantom

February 21, 2024

1 Genesis Analysis of Phillips phantom

This notebook demonstrates the use of the Phillips Distortion Phantom with an Elekta MRI-Unity. All analysed data is available here. Documentation for the `mri_distortion_toolkit` is here

The Phillips Phantom appears quite similar to the Image X phantom, consisting of parralel planes with embedded markers of some signal generating substrate. Compared to the sydney university phantom, it is much heavier, much more rigid, and has a much bigger field of view.

This analysis very closely follows the MrGoam analysis notebook, where I provide a fair bit of detail about what is going on - I suggest you read that one first so you understand the following steps! The image acquisition parameters were the same in each case.

1.1 Marker Extraction

One challenge with this phantom is that the markers don't have unique contrast on the CT. We were unable to easily extract the marker positions from the CT image, so instead we create an 'artificial' ground truth based on the known marker positions.

[3]:

```
import sys
import os
from pathlib import Path
this_file_loc = Path(os.path.abspath(''))
sys.path.insert(0, str(this_file_loc.parent))
from mri_distortion_toolkit.MarkerAnalysis import␣
  ↪MarkerVolume,MatchedMarkerVolumes
from python_codes.philips_phantom import philips_phantom
from mri_distortion_toolkit.utilities import plot_distortion_xyz_hist




# Distorted centroids corrected (will take around 10 minutes - these are big␣
  ↪images!)
mr_data_loc_with_DC = Path("/home/brendan/Dropbox (Sydney Uni)/Projects/
  ↪OpenDistortionPhantom/genesis_data/PhilipPhantom_MRI/2022-08__Studies/
  ↪GOAM^ImageX_ZZZIMAGEX_MR_2022-08-31_172341_._T2.3D.Tra.2min_n401__00002")
mri_volume_with_DC = MarkerVolume(mr_data_loc_with_DC)
```

```
this code has not been tested with non-siemens scanners. If dicom standards work
properlythis code will too…
total segmentation time:  639.5 s
```

[4]:

```
# Distorted Centroids uncorrected (will take a few minutes)
mr_data_loc_no_DC = Path("/home/brendan/Dropbox (Sydney Uni)/Projects/
 ↪OpenDistortionPhantom/genesis_data/PhilipPhantom_MRI/2022-08__Studies/
 ↪GOAM^ImageX_ZZZIMAGEX_MR_2022-08-31_172341_._T2.3D.Tra.2min_n401__00000")
mri_volume_no_DC = MarkerVolume(mr_data_loc_no_DC)
# mri_volume_no_DC.plot_3D_markers(title='MRI uncorrected')
```

```
this code has not been tested with non-siemens scanners. If dicom standards work
properlythis code will too…
total segmentation time:  564.2 s
```

[5]:

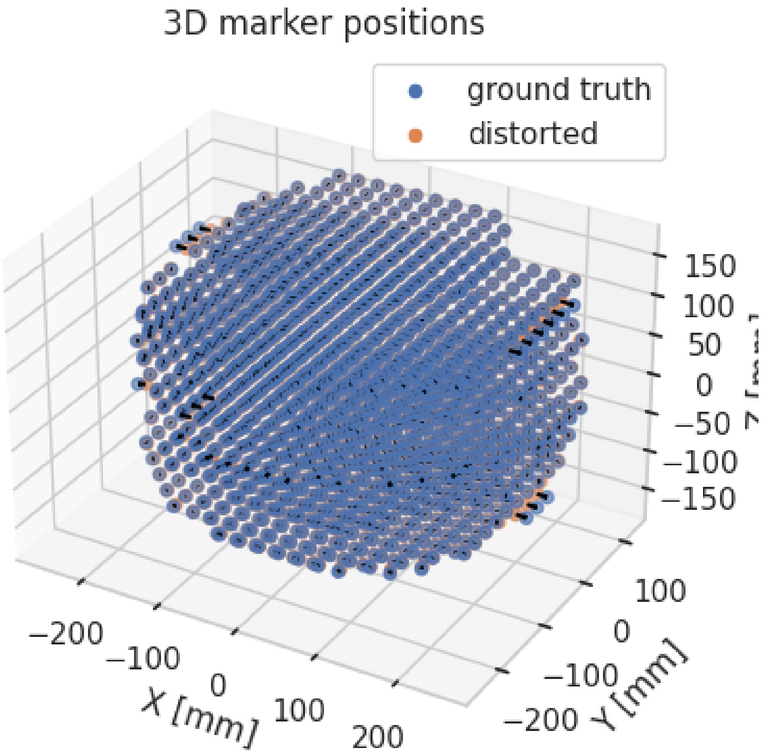
```
# Undistorted centroids

(x, y, z) = philips_phantom.get_alignment_coords(mri_volume_with_DC)
philips_centroids = philips_phantom.generate_centroids(x, y, z)
philips_gt = MarkerVolume(philips_centroids)
```

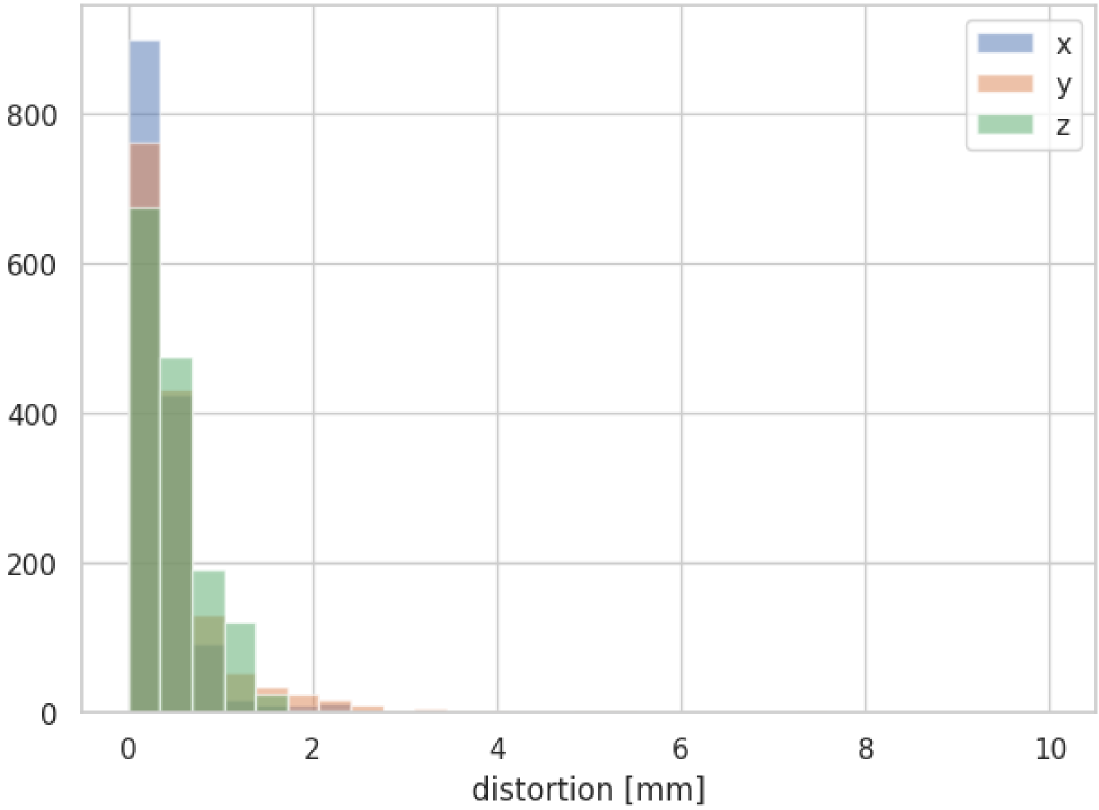
```
[MarkerAnalysis.py: line 132  WARNING] numpy input is deprecated, please use
pandas input with ['x', 'y', 'z'] cols instead
```

1.2 Marker Matching

[7]:

```
# # Match markers corrected
corrected = MatchedMarkerVolumes(philips_gt, mri_volume_with_DC)
corrected.plot_3D_markers()
plot_distortion_xyz_hist(corrected)
```

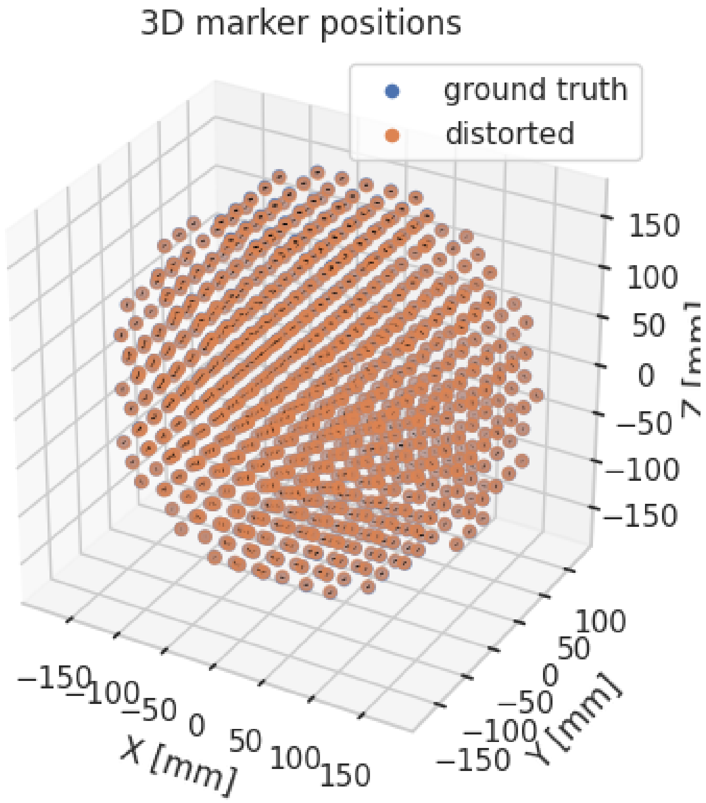

3D marker positions
● ground truth
● distorted
150
100
50
0
−50
−100
−150
Z [mm]
−200
−100
0
100
200
X [mm]
100
0
−100
−200
Y [mm]

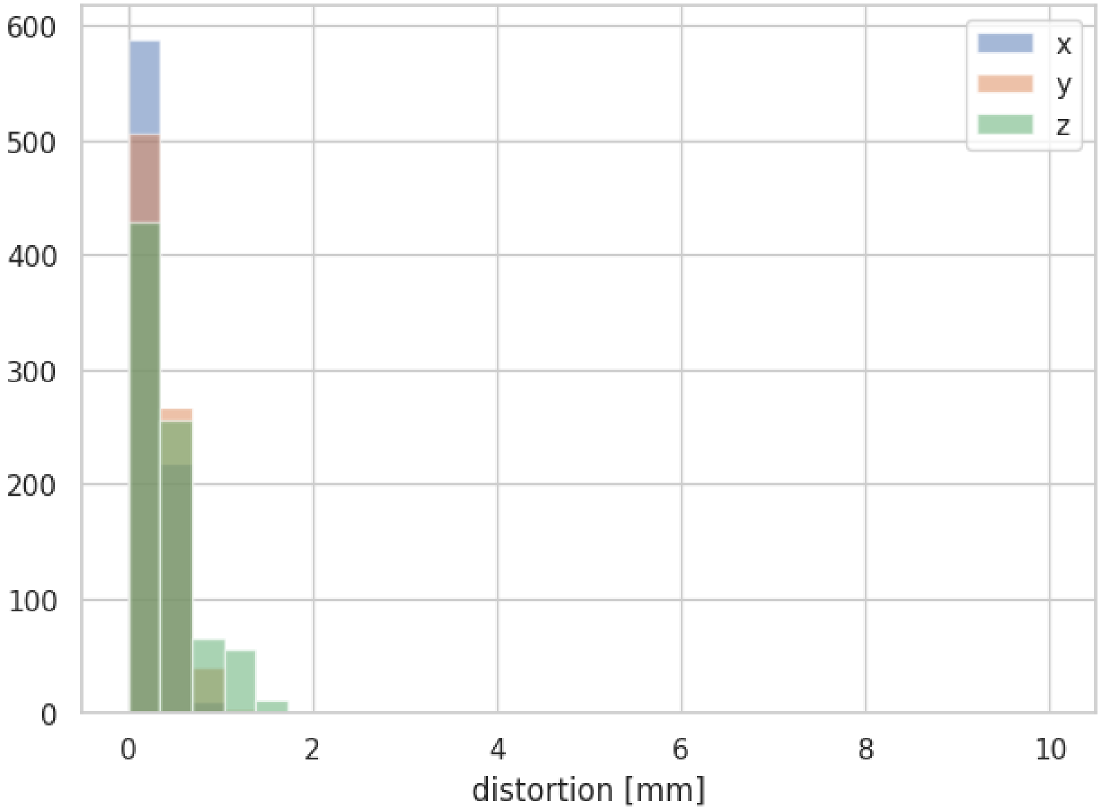


Note that these ‘markers’ on the edge are actually the result of phase wrapping artefacts. So the largest distortion we are seeing here are not real. We may be able to get rid of these artefacts by using a larger FOV, but for now we will get rid of them by just clipping the input data:

[8]:

```
mri_volume_with_DC.MarkerCentroids = mri_volume_with_DC.
  ↪MarkerCentroids[mri_volume_with_DC.MarkerCentroids['r'] < 190]
corrected = MatchedMarkerVolumes(philips_gt, mri_volume_with_DC)
corrected.plot_3D_markers()
plot_distortion_xyz_hist(corrected)
```


3D marker positions
ground truth
distorted
150
100
50
0
−50
−100
−150
Z [mm]
100
50
0
−50
−100
−150
Y [mm]
−150
−100
−50
0
50
100
150
X [mm]

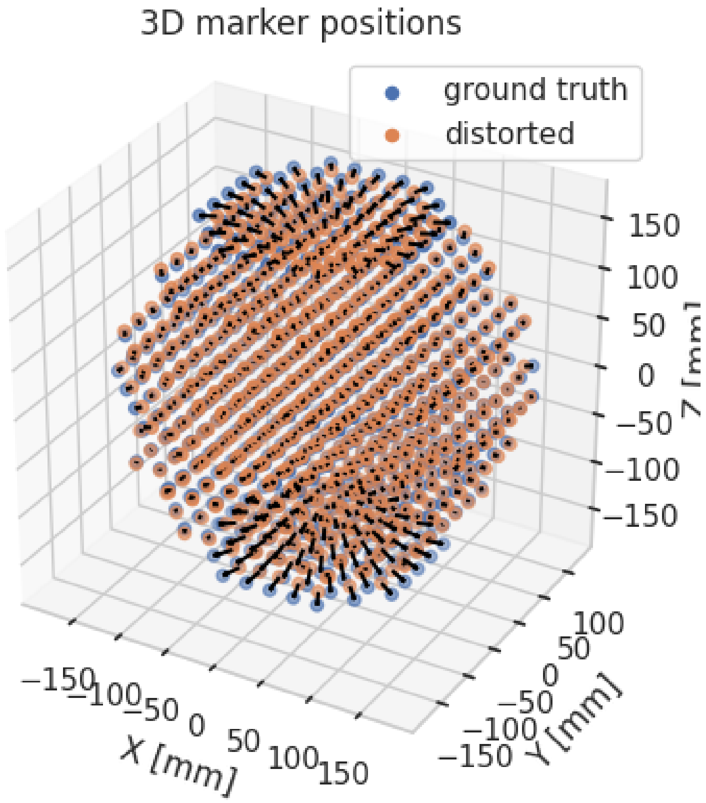


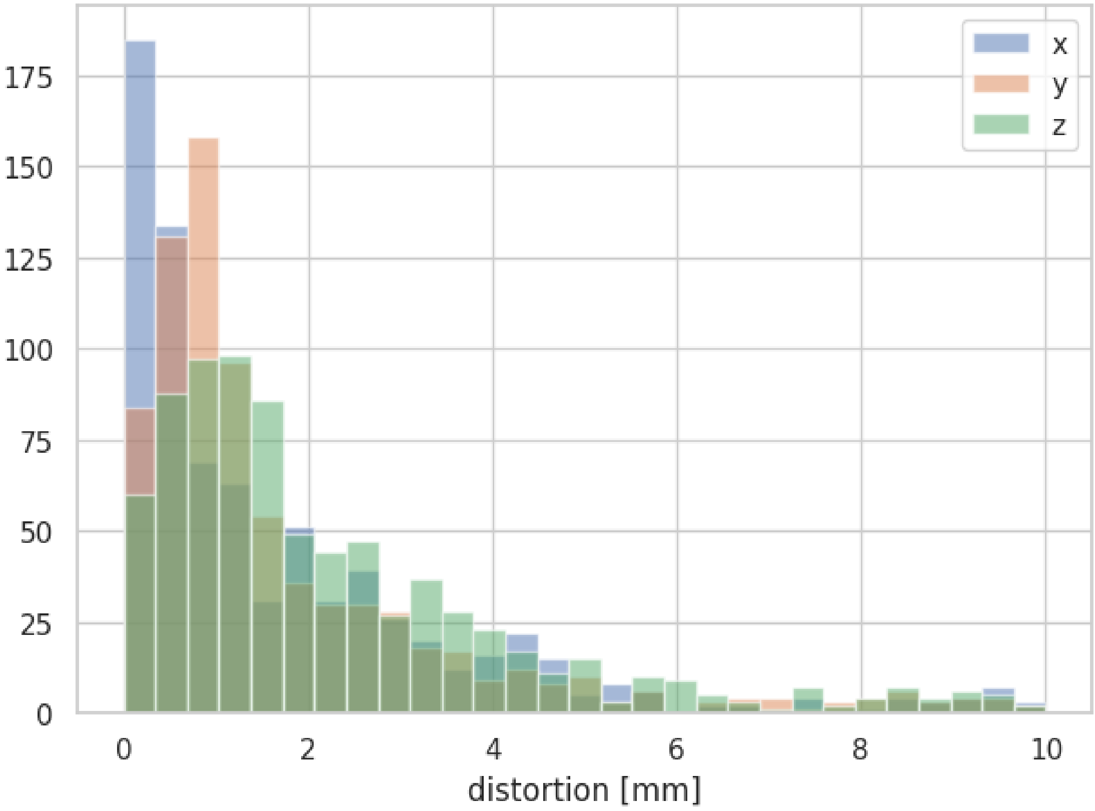
OK, once we've clipped the data to a total volume of 380 mm (still a very large FOV by our standards) all distortion is comfortably within 2 mm. This is quite impressive!

Let's take a look at the result without distortion correction. Again, we will first filter the markers by radius because otherwise we have phase wrap and marker overlap and it becomes impossible to perform a 1:1 match.

[9]:

```
# trim data:
mri_volume_no_DC.MarkerCentroids = mri_volume_no_DC.
  ↪MarkerCentroids[mri_volume_no_DC.MarkerCentroids['r'] < 190]
# match  data:
uncorrected = MatchedMarkerVolumes(philips_gt, mri_volume_no_DC)
uncorrected.plot_3D_markers()
plot_distortion_xyz_hist(uncorrected)
```


3D marker positions
ground truth
distorted
X [mm]
Y [mm]
Z [mm]
−150
−100
−50
0
50
100
150



We can also print ourselves out a little report:

```
[5]: print(f'Median distortion: {uncorrected.MatchedCentroids.match_distance.
      ↪median(): 1.1f} mm, '
            f'Max distortion: {uncorrected.MatchedCentroids.match_distance.max(): 1.
      ↪1f} mm')
```

```
Median distortion:  3.2 mm, Max distortion:  18.1 mm
```

So a maximum distortion of 18.1 over 380 mm with no distortion correction - honestly it's actually still pretty impressive.

1.3 Reporting - data driven

We can do this either using the data directly, or by first fitting harmonics and reconstructing the reporting data. The latter makes prettier figures, but you'd probably feel more confident using the data directly so we will do that. first I will start with the corrected data.

In addition, I am importing the Elekta_Distortion_tests. Users can also write their own test suite.

```
[29]: from mri_distortion_toolkit.Reports import MRI_QA_Reporter
      from mri_distortion_toolkit.Reports import Elekta_Distortion_tests
```

```
report = MRI_QA_Reporter(MatchedMarkerVolume=corrected.MatchedCentroids,
                         r_outer=190,
                         dicom_data=mri_volume_no_DC.dicom_data,
                         tests_to_run=Elekta_Distortion_tests)
report.write_html_report()
```

```
The report has been compiled and can be found at
/home/brendan/Documents/MR_QA_Reports
```

Note that by default the report will be written to "~/Documents/MR_QA_reports" but you can also pass `output_folder` and `report_name` to `write_html_reports`. The best way to look at such reports is on your system (or you can see an online example here, but to avoid any feelins of anti-climax, I'll copy one of the resultant plots directly into this notebook:

[30]:
```
from IPython.display import HTML
HTML('../_plots/distortion_v_r_21_02_2024_1.html')
```

[30]:
```
<IPython.core.display.HTML object>
```

1.4 Reporting - harmonic driven

The problem with the marker data is that it's relatively sparse, especially when it comes to generating pretty plots. Instead we can use harmonics to reconstruct data wherever we want. Of course, the two cases **should** produce very similar results!

[31]:
```
from mri_distortion_toolkit.FieldCalculation import ConvertMatchedMarkersToBz
from mri_distortion_toolkit import calculate_harmonics
import numpy as np

Bz = ConvertMatchedMarkersToBz(corrected.MatchedCentroids, mri_volume_no_DC.
 ↪dicom_data)
gradient_strength = np.array(mri_volume_no_DC.dicom_data['gradient_strength'])
normalisation_factor = [1 / gradient_strength[0], 1 / gradient_strength[1], 1 /␣
 ↪gradient_strength[2],
                        1]  # this normalised gradient harmonics to 1mT/m
G_x_Harmonics, G_y_Harmonics, G_z_Harmonics, B0_Harmonics =␣
 ↪calculate_harmonics(Bz.MagneticFields,
                                                                              ␣
 ↪n_order=5,
                                                                              ␣
 ↪scale=normalisation_factor)
```

```
Initial pk-pk:        3.996e+03  T
Reconstructed pk-pk:  3.994e+03  T
Residual pk-pk:       1.005e+01  T ( 0.3%)
Initial pk-pk:        2.678e+03  T
Reconstructed pk-pk:  2.678e+03  T
Residual pk-pk:       8.878e+00  T ( 0.3%)
```

```
Initial pk-pk:        3.754e+03  T
Reconstructed pk-pk:  3.751e+03  T
Residual pk-pk:       1.822e+01  T ( 0.5%)
```

Note the very low fit errors in each case; that is what we want to see

[32]:

```
report_harmonics = MRI_QA_Reporter(gradient_harmonics=[G_x_Harmonics.harmonics,
                                                       G_y_Harmonics.harmonics,
                                                       G_z_Harmonics.harmonics],
                                   r_outer=190,
                                   dicom_data=mri_volume_no_DC.dicom_data,
                                   tests_to_run=Elekta_Distortion_tests)
report_harmonics.write_html_report()
```

```
The report has been compiled and can be found at
/home/brendan/Documents/MR_QA_Reports
```

[28]:

```
from IPython.display import HTML
HTML('../_plots/distortion_v_r_21_02_2024.html')
```

[28]: `<IPython.core.display.HTML object>`

[ ]: