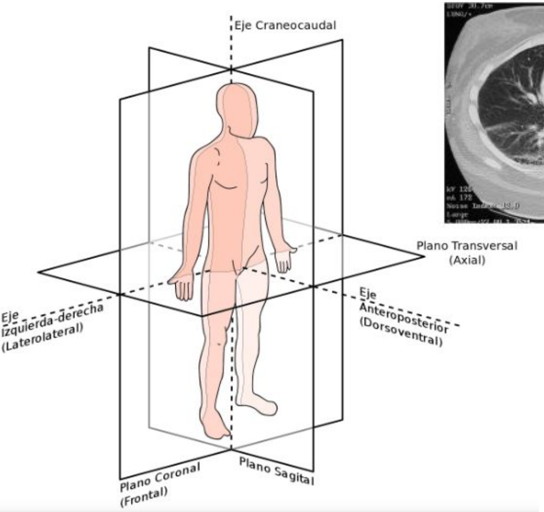
Final Lab

11763 - Procesamiento de Imágenes Médicas

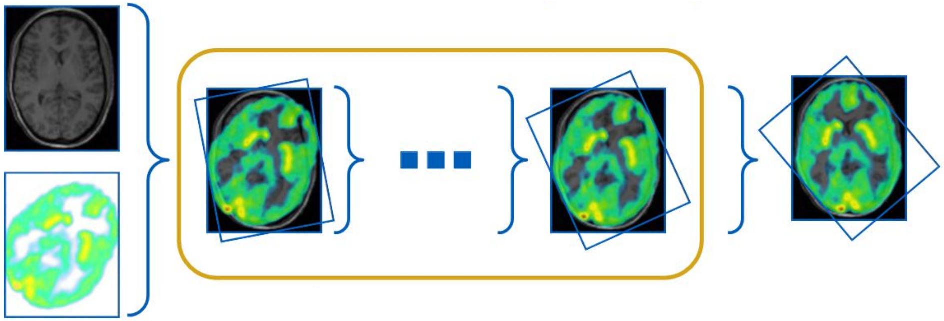
Martí Gelabert Gómez

What we have to do?

Task 1



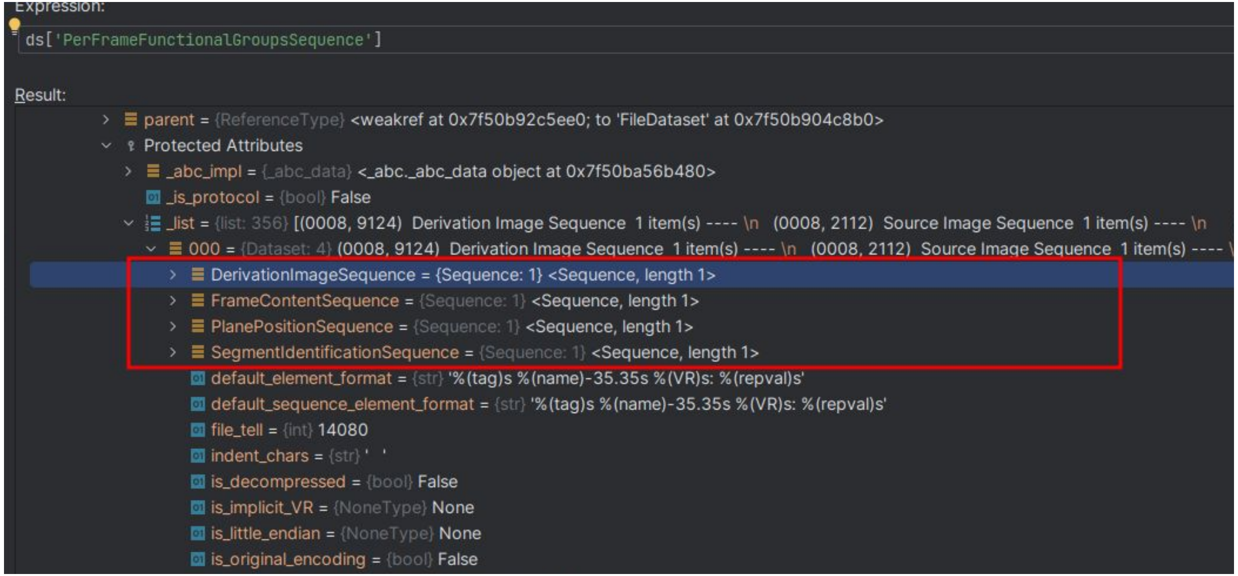
Task 2



Dicom Files

3D Slicer

Pixel Array
Slice Index
Position Patient



Task 1

Task 1: Procedure

1. Get information from DICOM files.
2. Order them with slice information.
3. Process Segmentation masks
4. Apply transformations to all.
5. Generate GIF

Task 1: Get the data

```python
ds = pydicom.dcmread('HCC_005/01-23-1999-NA-ABDPELVIS-36548/300.000000-Segmentation-06660/1-1.dcm')
segmentation = []
mask1 = []
mask2 = []
mask3 = []
mask4 = []

for index,i in enumerate(ds['PerFrameFunctionalGroupsSequence']):
    image_position = i.PlanePositionSequence[0].ImagePositionPatient # última componente

    segment_seq = i.SegmentIdentificationSequence
    if segment_seq is not None:
        segment_number = segment_seq[0].ReferencedSegmentNumber # esto te dice en que segmentación estas

        if segment_number == 1:
            mask1.append((ds.pixel_array[index], image_position[2]))

        if segment_number == 2:
            mask2.append((ds.pixel_array[index], image_position[2]))

        if segment_number == 3:
            mask3.append((ds.pixel_array[index], image_position[2]))

        if segment_number == 4:
            mask4.append((ds.pixel_array[index], image_position[2]))
```

```python
slices = []
for i in paths:
    ds = pydicom.dcmread(i)
    if hasattr(ds, 'SliceLocation'):
        slices.append(ds)
slices = sorted(slices, key=lambda s: s.SliceLocation)

# stack the slices to obtain a 89xIMG_SIZExIMG_SIZE
slides3d = np.array([s.pixel_array for s in slices])
```

```python
def sortMaks(mask):
    lista_ordenada = sorted(mask, key=lambda tupla: tupla[1])
    lista_A = [tupla[0] for tupla in lista_ordenada]
    return lista_A
```

Task 1: Apply Transformations

```python
5 usages  Martí Gelabert Gómez
def MIP_sagittal_plane(img_dcm: np.ndarray) -> np.ndarray:
    """ Compute the maximum intensity projection on the sagittal orientation. """
    return np.max(img_dcm, axis=2)
4 usages  Martí Gelabert Gómez
def MIP_coronal_plane(img_dcm: np.ndarray) -> np.ndarray:
    """ Compute the maximum intensity projection on the coronal orientation. """
    return np.max(img_dcm, axis=1)

5 usages  Martí Gelabert Gómez
def rotate_on_axial_plane(img_dcm: np.ndarray, angle_in_degrees: float) -> np.ndarray:
    """ Rotate the image on the axial plane. """
    return scipy.ndimage.rotate(img_dcm, angle_in_degrees, axes=(1, 2), reshape=False)
```

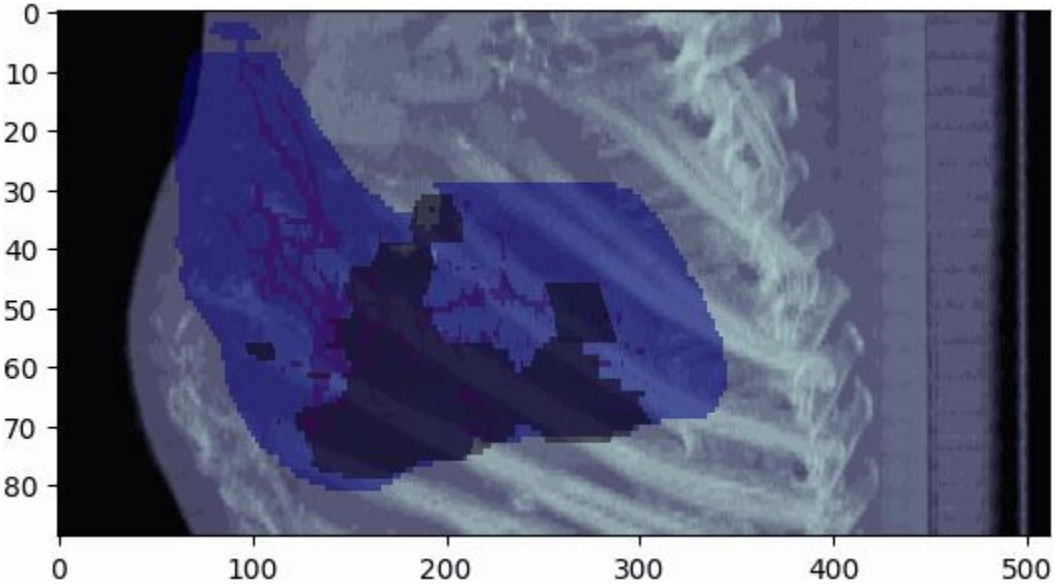
```python
n = 24
projections = []
for idx, alpha in enumerate(np.linspace(0, 360 * (n - 1) / n, num=n)):
    rotated_img = rotate_on_axial_plane(img_dcm, alpha)
    projection = MIP_sagittal_plane(rotated_img)

    rotated_mask1 = rotate_on_axial_plane(mask1, alpha)
    rotated_mask2 = rotate_on_axial_plane(mask2, alpha)
    rotated_mask3 = rotate_on_axial_plane(mask3, alpha)
    rotated_mask4 = rotate_on_axial_plane(mask4, alpha)
```

```python
mask1_projection = MIP_sagittal_plane(rotated_mask1)
mask2_projection = MIP_sagittal_plane(rotated_mask2)
mask3_projection = MIP_sagittal_plane(rotated_mask3)
mask4_projection = MIP_sagittal_plane(rotated_mask4)
```

Task 1: Show your projections

```
plt.imshow(projection, cmap=cm, vmin=img_min, vmax=img_max, aspect=pixel_len_mm[0] / pixel_len_mm[1])
plt.imshow(mask1_projection, cmap="jet", aspect=pixel_len_mm[0] / pixel_len_mm[1], alpha=0.5)
plt.imshow(mask2_projection, cmap="hot", aspect=pixel_len_mm[0] / pixel_len_mm[1], alpha=0.5)
plt.imshow(mask3_projection, cmap="viridis", aspect=pixel_len_mm[0] / pixel_len_mm[1], alpha=0.5)
plt.imshow(mask4_projection, cmap="inferno", aspect=pixel_len_mm[0] / pixel_len_mm[1], alpha=0.5)
```

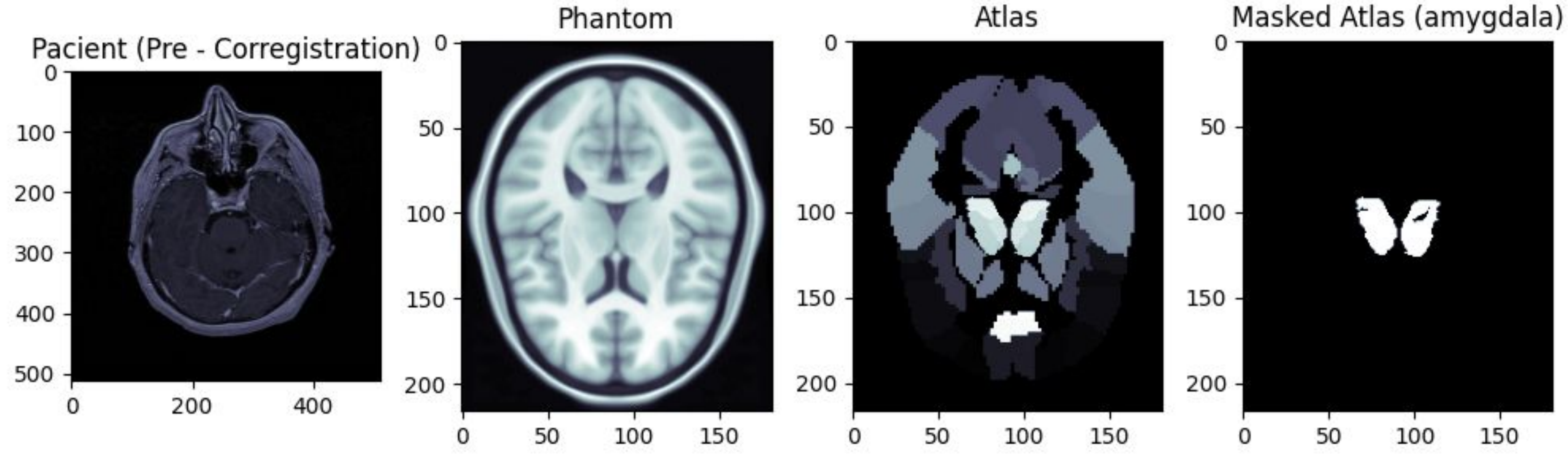

0
10
20
30
40
50
60
70
80
0
100
200
300
400
500

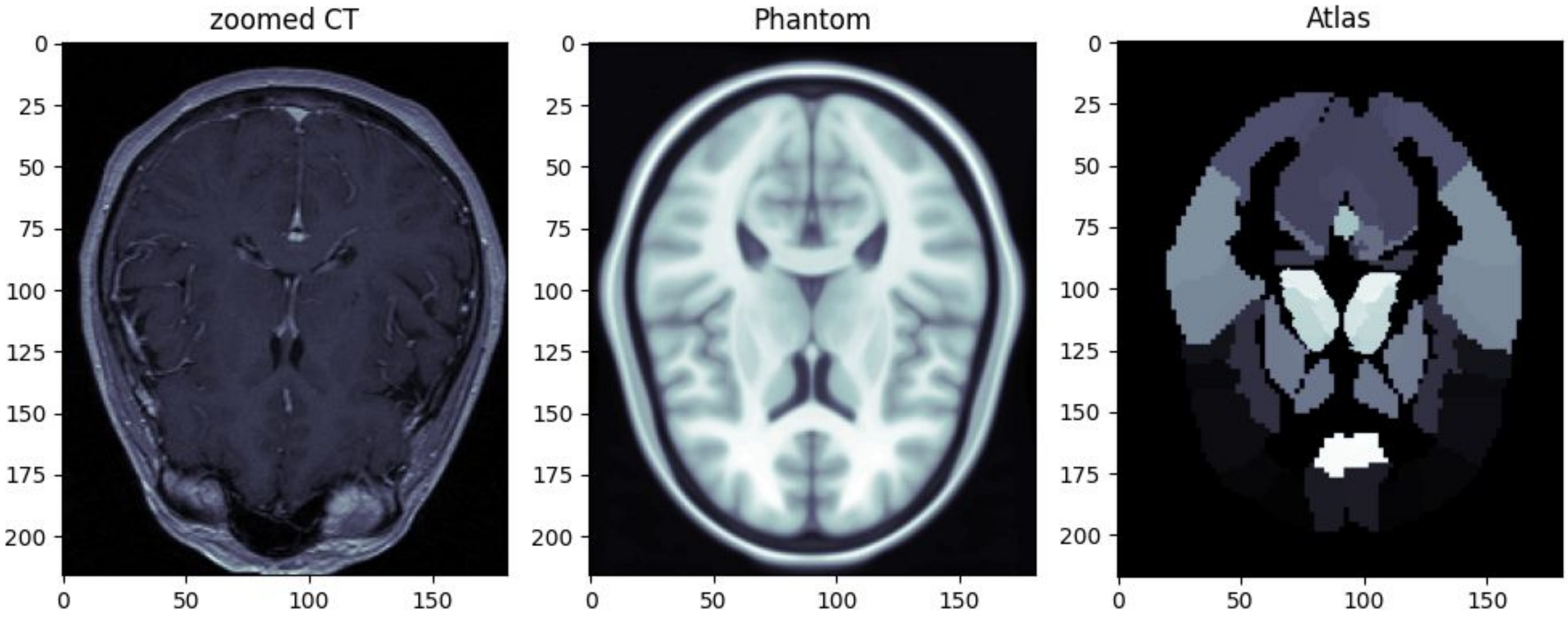
Task 2

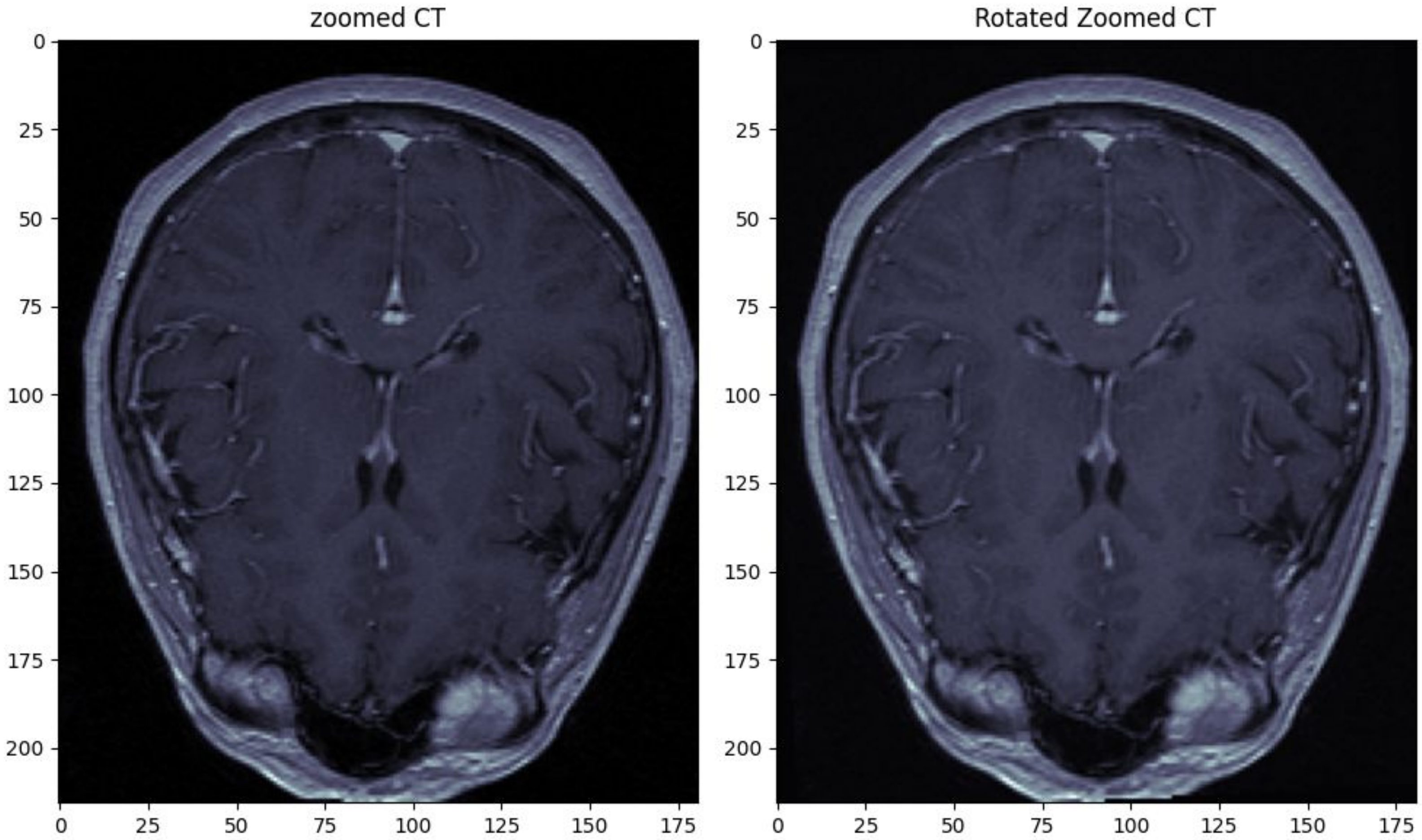
Task 2: Procedure

1. Get information from DICOM files.
2. Order them with slice information.
3. Fix CT proportions to match reference.
4. Find landmarks.
5. Use an optimization algorithm to find optimal parameters for transformation
6. Transform and apply with your optimal parameters.

Task 2: Fix proportions

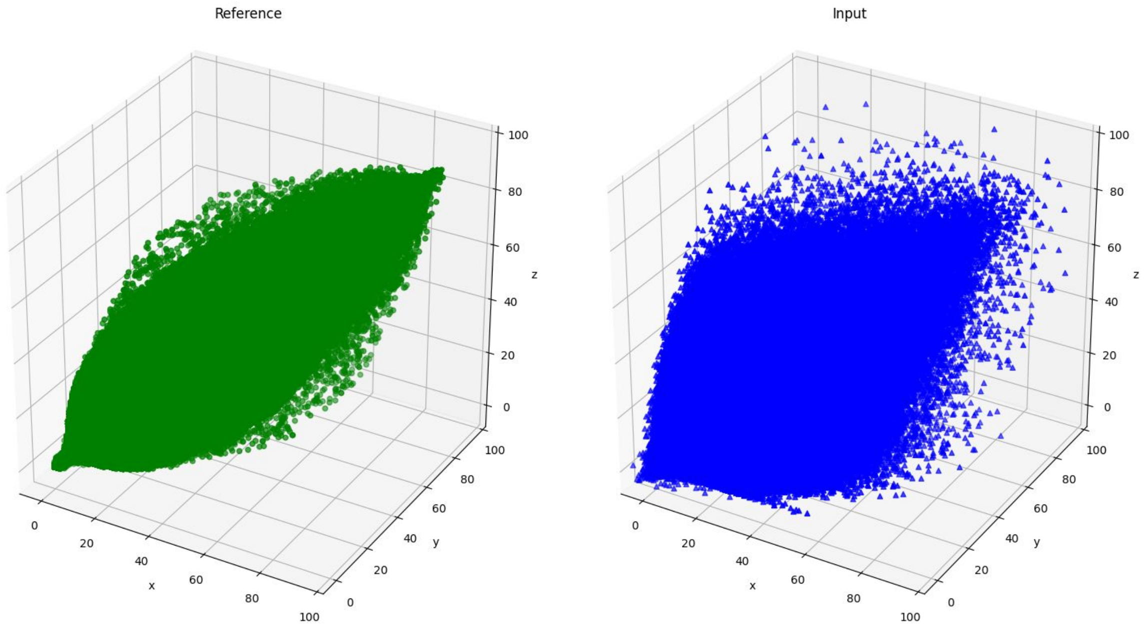



zoomed CT
Phantom
Atlas
0
25
50
75
100
125
150
175
200
0
50
100
150


zoomed CT
Rotated Zoomed CT
0
25
50
75
100
125
150
175
200
0
25
50
75
100
125
150
175

```
# Computing The landmarks (https://gist.github.com/tschreiner/8f971bbbd40606e58f1e4fb1852e8b8e)
landmarks_ref = images_phantom[::15, ::15, ::15].reshape(-1,3)
landmarks_input = processedCT[::15, ::15, ::15].reshape(-1,3)
```

Landmark Before coregistration



```python
def coregister_landmarks(ref_landmarks: np.ndarray, inp_landmarks: np.ndarray):
    """ Coregister two sets of landmarks using a rigid transformation. """
    initial_parameters = [
        0, 0, 0,  # Translation vector
        0,  # Angle in rads
        1, 0, 0,  # Axis of rotation
    ]
    # Find better initial parameters
    centroid_ref = np.mean(ref_landmarks, axis=0)
    centroid_inp = np.mean(inp_landmarks, axis=0)
    # Your code here:
    #   ...
    initial_parameters[0] = centroid_ref[0] - centroid_inp[0]
    initial_parameters[1] = centroid_ref[1] - centroid_inp[1]
    initial_parameters[2] = centroid_ref[2] - centroid_inp[2]

    Martí Gelabert Gómez
    def function_to_minimize(parameters):
        """ Transform input landmarks, then compare with reference landmarks."""
        # Your code here:
        #   ...
        inp_landmarks_transf = np.asarray(
            [translation_then_axialrotation(point, parameters) for point in inp_landmarks])
        return vector_of_residuals(ref_landmarks, inp_landmarks_transf)

    # Apply least squares optimization
    result = least_squares(
        function_to_minimize,
        x0=initial_parameters,
        verbose=1)
    return result
```

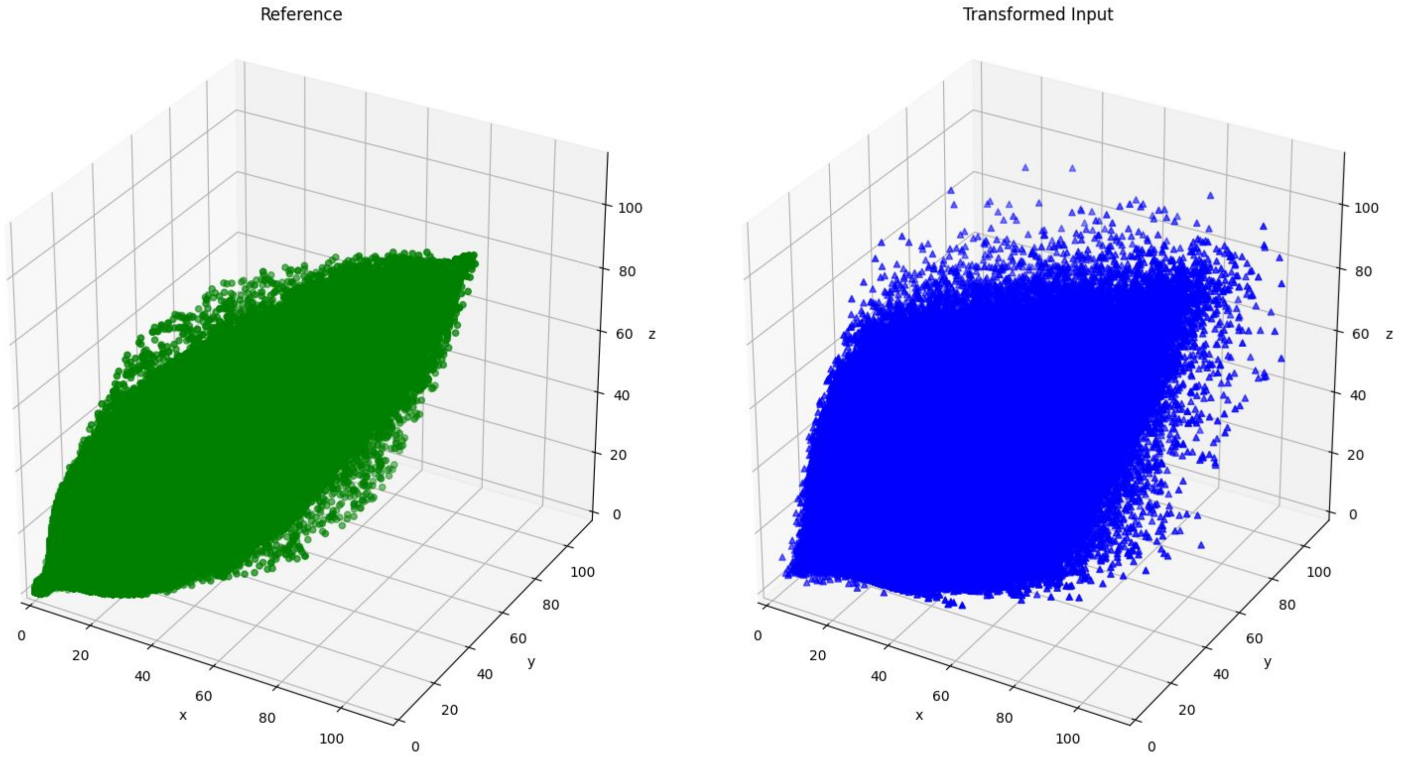
>> Mean residual value: 24.97651824358491. (before corregistration)

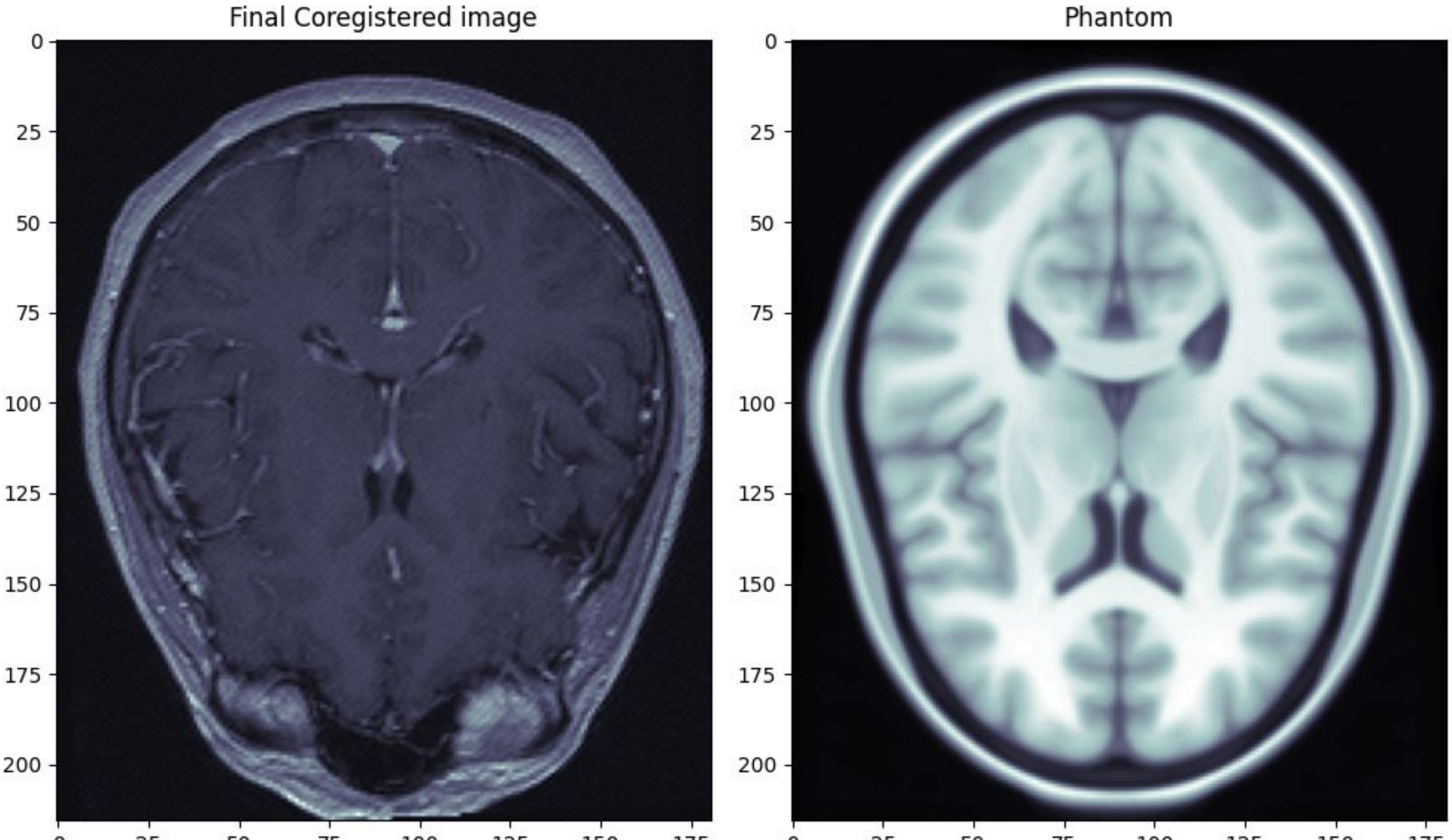
>> Best parameters: ([7.76592284 8.37130049 10.05798676 -0.25040872 0.59583415 0.44332766 0.71613931]

mean_absolute_error (centroid_idx)>> 17.34841054272961

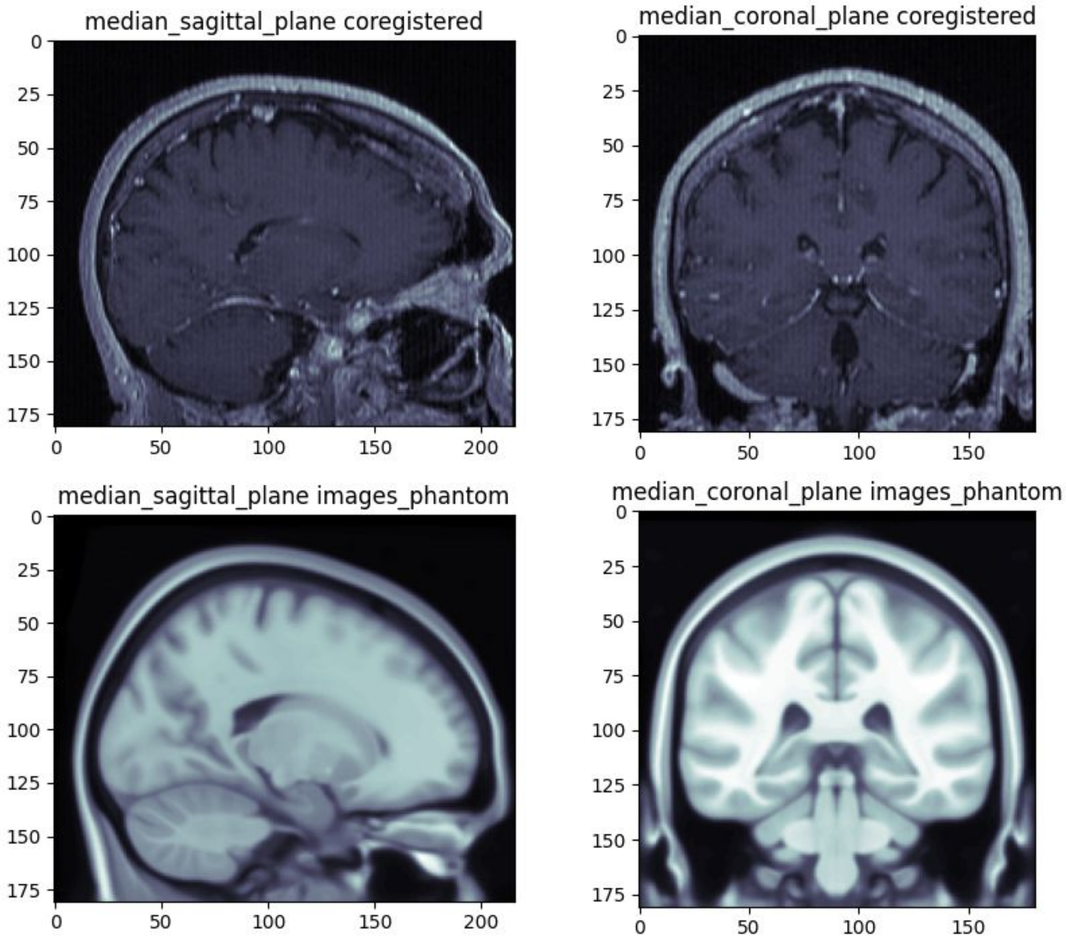
mean_squared_error (centroid_idx)>> 433.00956087370577

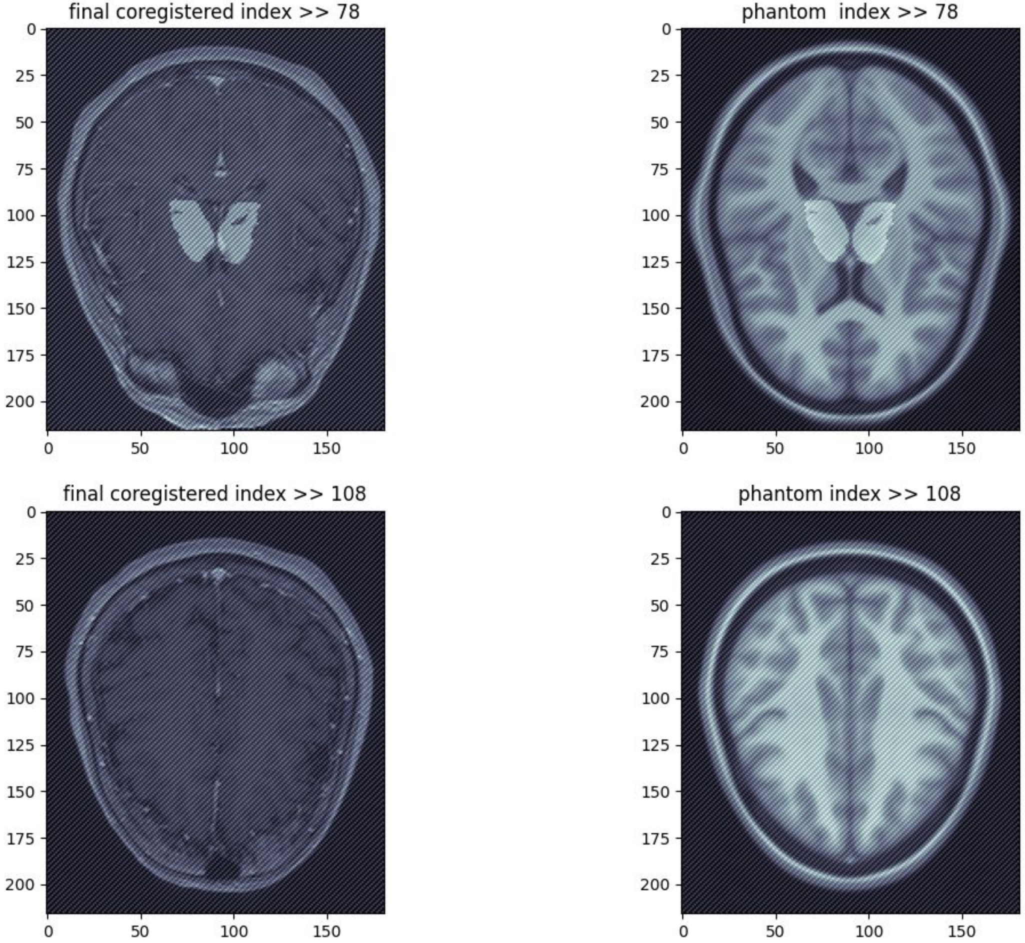
mutual_information (centroid_idx)>> 0.6591496641902275


Reference
Transformed Input
x
y
z
0
20
40
60
80
100



Final Coregistered image
Phantom
0
25
50
75
100
125
150
175
200


median_sagittal_plane coregistered
median_coronal_plane coregistered
median_sagittal_plane images_phantom
median_coronal_plane images_phantom


final coregistered index >> 78
phantom index >> 78
final coregistered index >> 108
phantom index >> 108
0
25
50
75
100
125
150
175
200
0
50
100
150

Q&A

Thank you for your time!