Subject

The `Subject` is a data structure used to store images associated with a subject and any other metadata necessary for processing.

All transforms applied to a `Subject` are saved in its `history` attribute (see Reproducibility).

```
class torchio.Subject(*args, **kwargs: Dict[str, Any])
```
[source]

Bases: `dict`

Class to store information about the images corresponding to a subject.

PARAMETERS

- ***args** – If provided, a dictionary of items.
- ****kwargs** – Items that will be added to the subject sample.

EXAMPLE

```
>>> import torchio as tio
>>> # One way:
>>> subject = tio.Subject(
...     one_image=tio.ScalarImage('path_to_image.nii.gz'),
...     a_segmentation=tio.LabelMap('path_to_seg.nii.gz'),
...     age=45,
...     name='John Doe',
...     hospital='Hospital Juan Negrín',
... )
>>> # If you want to create the mapping before, or have spaces in the keys:
>>> subject_dict = {
...     'one image': tio.ScalarImage('path_to_image.nii.gz'),
...     'a segmentation': tio.LabelMap('path_to_seg.nii.gz'),
...     'age': 45,
...     'name': 'John Doe',
...     'hospital': 'Hospital Juan Negrín',
... }
>>> subject = tio.Subject(subject_dict)
```

```
add_image(image: torchio.data.image.Image, image_name: str) → None
```
[source]

Add an image.

```
apply_inverse_transform(**kwargs) → torchio.data.subject.Subject
```
[source]

Try to apply the inverse of all applied transforms, in reverse order.

PARAMETERS

****kwargs** – Keyword arguments passed on to `get_inverse_transform()`.

```
check_consistent_attribute(attribute: str, relative_tolerance: float = 1e-06,
    absolute_tolerance: float = 1e-06, message: Optional[str] = None) → None
```
[source]

Check for consistency of an attribute across all images.

PARAMETERS

- **attribute** – Name of the image attribute to check
- **relative_tolerance** – Relative tolerance for `numpy.allclose()`
- **absolute_tolerance** – Absolute tolerance for `numpy.allclose()`

EXAMPLE

```
>>> import numpy as np
>>> import torch
>>> import torchio as tio
>>> scalars = torch.randn(1, 512, 512, 100)
>>> mask = torch.tensor(scalars > 0).type(torch.int16)
>>> af1 = np.eye([0.8, 0.8, 2.50000000000001, 1])
>>> af2 = np.eye([0.8, 0.8, 2.49999999999999, 1])  # small difference here (e.g. due
>>> subject = tio.Subject(
...   image = tio.ScalarImage(tensor=scalars, affine=af1),
...   mask = tio.LabelMap(tensor=mask, affine=af2)
... )
>>> subject.check_consistent_attribute('spacing')  # no error as tolerances are > 0
```

Note

To check that all values for a specific attribute are close between all images in the subject, `numpy.allclose()` is used. This function returns `True` if $|a_i - b_i| \leq t_{abs} + t_{rel} * |b_i|$, where a_i and b_i are the i-th element of the same attribute of two images being compared, t_{abs} is the `absolute_tolerance` and t_{rel} is the `relative_tolerance`.

```
get_inverse_transform(warn: bool = True, ignore_intensity: bool = True,
    image_interpolation: Optional[str] = None) → Compose
```
[source]

Get a reversed list of the inverses of the applied transforms.

PARAMETERS

- **warn** – Issue a warning if some transforms are not invertible.
- **ignore_intensity** – If `True`, all instances of `IntensityTransform` will be ignored.
- **image_interpolation** – Modify interpolation for scalar images inside transforms that perform resampling.

```
load() → None
```
[source]

Load images in subject on RAM.

```
plot(**kwargs) → None
```
[source]

Plot images using matplotlib.

PARAMETERS

****kwargs** – Keyword arguments that will be passed on to `plot()`.

```
remove_image(image_name: str) → None [source]
```

Remove an image.

```
property shape
```

Return shape of first image in subject.

Consistency of shapes across images in the subject is checked first.

Example:

```
>>> import torchio as tio
>>> colin = tio.datasets.Colin27()
>>> colin.shape
(1, 181, 217, 181)
```

```
property spacing
```

Return spacing of first image in subject.

Consistency of spacings across images in the subject is checked first.

Example:

```
>>> import torchio as tio
>>> colin = tio.datasets.Slicer()
>>> colin.shape
(1.0, 1.0, 1.2999954223632812)
```

```
property spatial_shape
```

Return spatial shape of first image in subject.

Consistency of spatial shapes across images in the subject is checked first.

Example:

```
>>> import torchio as tio
>>> colin = tio.datasets.Colin27()
>>> colin.shape
(181, 217, 181)
```

Previous
Image

Next
Dataset