SNAKE-fMRI: A modular fMRI data simulator
from the space-time domain to k-space and back.
Supplementary material.

A Interacting with SNAKE-fMRI

A.1 Create your custom handler

Due to the modularity of Snake-fMRI it is easy to add your own modeling steps. Here we propose a simple example that adds scanner drift to the acquired data.

```python
import numpy as np
from nilearn.glm.first_level.design_matrix import _make_drift
from snkf.handlers import AbstractHandler, requires_field

@requires_field("data_acq")  # ensure that simulation has the required data.
class PolynomialScannerDriftHandler(AbstractHandler):
    """Add Polynomial drift to the data."""

    __handler_name__ = "scanner-polydrift"

    # parameters for the Handler, dataclass-like definition.
    drift_order: int
    drift_intensities: np.ndarray

    def _handle(self, sim):
        # Nilearn does the heavy lifting
        frames_tr =np.linspace(0, sim.sim_time, sim.n_frames)
        drift_matrix = _make_drift("polynomial", frame_times=frames_tr, order=self._drift_order)
        drift_matrix = drift_matrix[:, :-1] # remove intercept column
        drift_intensity = np.linspace(1, 1 + self.drift_intensities, sim.n_frames)
        timeseries = drift_intensity @ drift_matrix
        sim.data_acq[:, sim.static_vol > 0] *= timeseries[:, np.newaxis] # apply drift
        return sim
```

A.2 Declaring a new simulation

Now we can create a new simulation using our handler. More examples of handlers usage are available in *SNAKE-fMRI* documentation.

```python
from snkf.simulation import SimData
from my_local_package import ScannerDriftHandler
from snkf.handlers import H

sim = SimData(shape=(64,64), fov=(.192, .192), sim_time=300, sim_tr= 0.1, )
simulator = H["phantom-big"] >> H["activation-block"] >> H["scanner-poly-drift"]
sim = simulator(sim)  # update the simulation by running it through the handlers.
```

A simulation can also be described using a configuration file, using `hydra` conventions:

```yaml
defaults:
  - handlers:
    - phantom-brainweb
    - activation-block
    - noise-gaussian
    - acquisition-vds
  - reconstructors: adjoint
  - _self_

cache_dir: ${oc.env:PWD}/cache

sim_params:
  sim_tr: 0.1       # time resolution in image domain (s)
  sim_time: 300     # total time of experiments
  shape: [-1,-1,-1] # inherited from phantom
  fov: [-1,-1,-1]   # inherited from phantom.
  n_coils: 1        # single coil for fast computations
  rng: 19980408     # random seed
  lazy: True        # Use the lazy generation of volume.

handlers:
  phantom-brainweb: # Create the phantom
    sub_id: 5
    bbox: [0.225,-0.07, 0.06, -0.055, null, null] # reduce the FOV to exclude spine
    brainweb_folder: ${cache_dir}/brainweb
    res: [3.0, 3.0, 3.0] # resolution in mm
  activation-block: # Add bold signal
    event_name: block_on
    block_on: 20
    block_off: 20
    duration: 300
    bold_strength: 0.02
  noise-gaussian: # add gaussian noise
    snr: 10
  acquisition-vds: # perform acquisition, fully sampled is a special case of VDS sampling.
    shot_time_ms: 50
    acs: 1
    accel: 1
    accel_axis: -1
    constant: true
    order: TOP_DOWN
    smaps: false

reconstructors:
  adjoint: {} #  Simple FFT reconstruction.

stats:
  contrast_name: ${handlers.activation-block.event_name}
```

All configuration files for running the scenarios described in the manuscript are available in the snake-fmri repository. After installation they can be run as follows:

```
  $ pip install snake-fmri
  $ snkf-main --config-name="scenario1"
  # Using Hydra, parameters can be modified and run over a grid of parameter.
  $ snkf-main --config-name="scenario2" -m ++reconstructors.sequential.restart_strategy=cold,warm,refine
```

B Study of SNR impact in Scenario 1

To help analyse the impact of input SNR (SNR_i) on the results we performed simulation with a sweep of different SNR values as well as the number of coils in the setup of Scenario S1. The reconstruction being a simple linear operation (Smaps combination and Inverse fourier transform) we can directly see the impact on the reconstruction.

The addition of uniform gaussian noise in the image results in the degradation of image quality metrics, in a linear fashion (Figure 1). For $tSNR$, the relation follows the same principle. Overall, setting $SNR_i = 10$ as in our scenarios, provides a realistic $SNR_{\text{exp}} \simeq 40$ and $tSNR \simeq 30$. This reproduces a thermal noise dominant setting, which particularly occurs at high resolution.

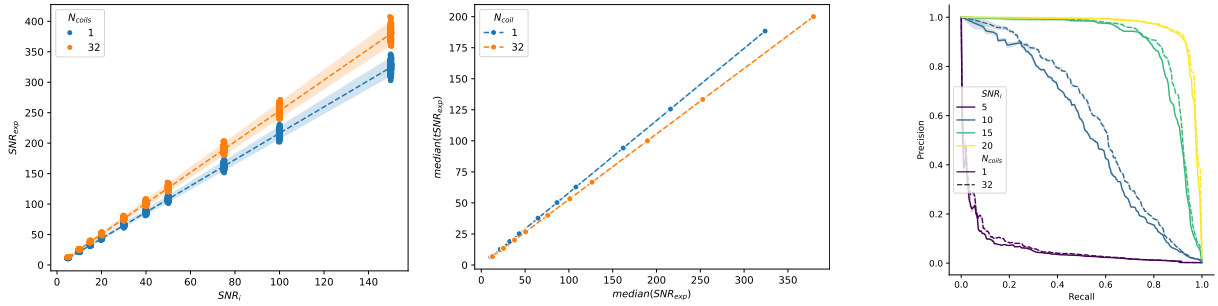

Figure 1: Impact of input SNR in Scenario 1 on image quality and statistics.

C Non-Cartesian trajectories design and expansion

By combining SNAKE-fMRI with MRI-NUFFT, it is possible to create and explore non-Cartesian trajectories, and small example of base trajectories (2D shots) and expansions (stack, rotate, conifying) are represented on Figure 2.

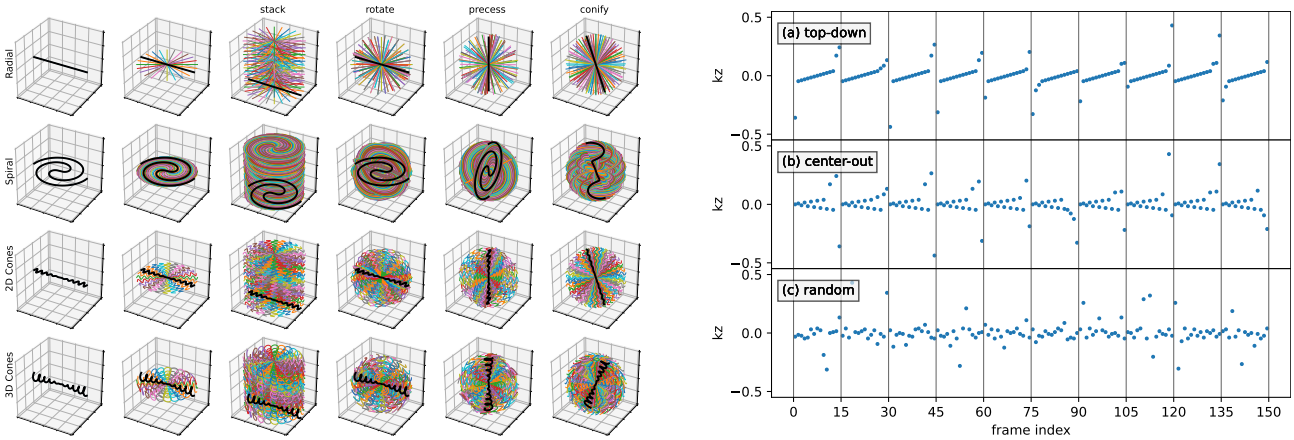

Figure 2: Left: Overview of trajectories generation possible with MRI-NUFFT. Right: Further expansions available in *SNAKE-fMRI*over the time axis are also possible, notably in which order the shot of each k-space frames are acquired, here for a stacked acquisition