## INSTALLATION

1 Welcome to the official PyDesigner project! 3
2 Notable Features 5
3 References 7
4 Indices and tables 33
WELCOME TO THE OFFICIAL PYDESIGNER PROJECT!

PyDesigner was inspired by NYU’s DESIGNER dMRI preprocessing pipeline to bring pre- and post-processing to every MRI imaging scientist. With PyDesigner, users are no longer confined to specific file types, operating systems, or complicated scripts just to extract DTI or DKI parameters – PyDesigner makes this easy, and you will love it!
CHAPTER TWO

NOTABLE FEATURES

- **100% Python-based** scripts
- **Minimized package dependencies** for small package footprint
- Preprocessing designed to **boost SNR**
- **Accurate and fast** DTI and DKI metrics via cutting-edge algorithms
- **One-shot** preprocessing to parameter extraction
- **Cross-platform compatibility** between Windows, Mac and Linux using Docker
- Highly flexible and **easy to use**
- **Parallel processing** for quicker preprocessing and parameterization
- **Easy install** with `pip`
- Input **file-format agnostic** – works with .nii, .nii.gz, .mif and dicoms
- **Quality control metrics** to evaluate data integrity – SNR graphs, outlier voxels, and head motion
- Uses the **latest techniques** from DTI/DKI/FBI literature
- Works with DTI, DKI, WMTI, FBI, or FBWM datasets

We welcome all DTI/DKI researchers to evaluate this software and pass on their feedback or issues through the Issues page of this project’s GitHub repository. Additionally, you may join the M-AMA Slack channel for live support.

**System Requirements** Parallel processing in PyDesigner scales almost linearly with the number of CPU cores present. The application is also memory-intensive due to the number of parameter maps being computed.

Based on this evaluation, for processing a single DWI using PyDesigner, we recommend the following minimum system specifications:

- Ubuntu 18.04
- Intel i7-9700 or AMD Ryzen 1800X [8 cores]
- 16 GB RAM
- 12 GB free storage
- Nvidia CUDA-enabled GPU
The PyDesigner software packages is based upon the references listed below. Please be sure to cite them if PyDesigner was used in any publications.


### 3.1 Changelog

All notable changes to this project will be documented in this file or page.
3.1.1 v1.0-RC8

Feb 15, 2021

Added:

- Added missing Rician preprocessing to -s, --standard preprocessing

Changed

- Potential sources of errors in FBWM have been mitigated with error-handling

Removed

- None

3.1.2 v1.0-RC7

Feb 11, 2021

Added:

- Missing Docker figures in RTD documentation

Changed

- Added error mitigation when FBI cost function fails to converge to a minimum cost
- Updated WMTI calculation to follow DKE outputs

Removed

- Unnecessary WMTI calculations

3.1.3 v1.0-RC6

Dec 22, 2020

Added:

- None

Changed

- Replaced `preprocessing.util.bvec_is_fullsphere()` and `preprocessing.util.vecs_are_fullsphere()` with `preprocessing.mrinfoutil.is_fullsphere()`. Even though datasets may be half-shelled, it is inaccurate to label them as such because distortion relative to b-value is not linear. As such, the `slm=linear` makes no sense. This new method performs the proper checks required before labelling a DWI as fully-shelled. A DWI is half-shelled iff max B-value is less than 3000 AND the norm of the mean direction vector is more than 0.3.

Removed

- See above
3.1.4 v1.0-RC5

Oct 26, 2020

Added:
  • Check for b-value scaling so .bval file so values specified as either 2.0 or 2000 can be processed.
  • fitting.dwipy() can now be pointed to user-defined bvec and bval paths. It previously required bvec and
    bval files to have the same name and path as DWI.
  • DSI Studio tractography for FBI. Processing FBI dataset now produces an fbi_tractography_dsi.fib
    file that can be loaded into DSI Studio to perform tractography.

Changed:
  • Fixed issue where eddy correction would attempt
to QC and fail despite parsing the --noqc flag.
  • SNR plotting works in very specific scenarious when input DWIs are of the same same dimensions. A try/except
    loop now ensure that the entire pipeline doesn’t halt due to errors in plotting.

Removed:
  • None

3.1.5 v1.0-RC4

Sep 22, 2020

Added:
  • Reslicing compatibility udpated for new MRTrix3 version where mrrelice has been changed to mrgrid.
    PyDesigner will work with either versions.

Changed:
  • Fixed a bad indent in tensor reordering function that produced an error in DTI protocols.

Removed:
  • None

3.1.6 v1.0-RC3

Sep 21, 2020

Added: * FBI fODF map for FBI tractography. Users may use MRTrix3
to further process this file.
  • Variable maximum spherical harmonic degree to improve robustness of FBI fit. This was fixed at 6 previous, but
    has been defaulted to 6 now. Users may change l_max with the -l_max n flag. This is based on information
    found at https://mrtrix.readthedocs.io/en/dev/concepts/sh_basis_lmax.html

Changed:
  • None

Removed:
  • None

3.1. Changelog 9
3.1.7 v1.0-RC2

Aug 25, 2020

Added:

• References to README.rst

Changed:

• The minimum B-value required for FBI (4000) is now inclusive instead of exclusive. This would allow execution of FBI/FBWM for datasets with $b=4000\ mm/s^2$
• Convert variable `nthreads` to string so `subproces.run` can recognize the flag
• Updated Slack permalink in README.rst

Removed:

• None

3.1.8 v1.0-RC1

Aug 19, 2020

Added:

• Methods to perform tensor only with compatible B-values. PyDesigner previously use all B-values in a DWI to do so. This behavior has been updated to use only B-values less than 2500
• FBI and FBWM calculations
• Brief documentation on how to run PyDesigner

Changed:

• Automatically issues `dwipreproc` or `dwifslpreproc` for compatibility with MRtrix3 $\geq 3.0.1$
• Updated minimum version for required Python modules

Removed:

• None

3.1.9 v0.32

Apr 21, 2020

Added:

• Intrinsic inter-axonal and mean extra-axonal diffusivity calculation to WMTI

Changed:

• Method `json2fslgrad` converted from class method to function definition
• `json2fslgrad` now transposes B0s in BVAL file in accordance with FSL’s gradient scheme
• Documentation update
• Extras directory renamed to extras
• DKE conversion scripts modified to correctly create ft and dke parameter files

Removed:
3.110 v0.31

Apr 9, 2020

Added:

• NaN check in AWF calculation that prevents further errors in intra-axonal and extra-axonal WMTI metrics computation

Changed:

• `designer.fitting.dwipy` input file detection method
• `Dockerfile_release` now deletes the correct temporary file to prevent build error

Removed:

• None

3.111 v0.3

Apr 8, 2020

Added:

• Head motion plot from `eddy_qc` outputs
• Outlier plot from IRRLS outlier detection
• Updated documentation
• Option to reslice DWI with `--reslice [x,y,z]`

Changed:

• Flag `--epiboost [index]` changed to `--epi [n]`, where users can specify the number of reverse phase encoded B0 pairs to use in EPI correction. Non-indexed B0s were previously destructively removed from DWI, leading to incorrect weighing of B0s in tensor estimation. The new method now preserves all B0s, thereby allowing faster EPI distortion correction without degrading DTI/DKI maps.
• Documentation moved to ReadTheDocs
• Moved B0 production module from `designer.preprocessing.brainmask` to a separate function at `designer.preprocessing.extractmeanbzero()` that gets called by PyDesigner main. This allows a B0.nii to be produced regardless of the `--mask` flag.

Removed:

• Documentation inconsistencies
3.1.12 v0.2 [The Cupid Release]

Feb 26, 2020

Added:

- Installer for setup with `pip install .`
- Multiple file support: `.nii`, `.nii.gz`, `.dcm`, `.mif`
- reStructuredText styled documentation
- Ability to use `--resume` flag for DWI concatenation
- SNR plot to depict signal changes before and after preprocessing
- Full utilization of AVX instruction set on AMD machines
- WMTI parameters

Changed:

- Fixed topup series not being denoised

Removed:

- CSF masking; feature failed to work consistently

3.1.13 dev-0.11

Dec 2, 2019

Added:

- None

Changed:

- Fixed bug in Dockerfile that prevented `pydesigner.py` from being found

Removed:

- None

3.1.14 0.1-dev

Oct 22, 2019

Initial port of MATLAB code to Python. 200,000,000,000 BCE

3.2 PyDesigner Requirements

PyDesigner, currently, only requires the following three dependencies:

1. Python 3.6, or above
2. FSL 6.0.2, or above
3. MRtrix3, 3.0_RC3 or above
3.2.1 Linux and Mac Users

Unix-based system users are able to natively run all dependencies. Please proceed with the installation steps to configure PyD.

3.2.2 Windows Users

FSL and MRtrix3 are currently not available on the Microsoft Windows platform. Users running Windows are recommended to run the Docker image NeuroDock these interdependencies at near-native speed.

You may still proceed with the installation of PyDesigner Python modules to perform tensor fitting and map extraction.

3.3 FSL

FSL is a collection of tools and software used to process fMRI, MRI and DWI data. Visit their installation page for download and installation guide.

**FSL 6.0.2 and above are recommended.** All testing has been done with FSL 6.0.2. PyDesigner has not been tested with other versions of FSL.

To check your FSL version:

```
$ flirt -version
```

A return value of at least FLIRT version 6.0 indicates successful installation of FSL, and that meets the PyD requirement.

3.4 MRtrix3 Installation

MRTRIX3 is another software suite aimed at analysis of DWI data. Here are some of their helpful pages.

1. Homepage
2. Download and Install

Confirm the success of installation with `mrinfo -version`. A valid output indicates successful installation.

3.5 Python

PyDesigner was built and tested on Python 3.7, so we encourage all users to adopt this version as well. While you may use the Python supplied by default on your OS, we highly encourage users to adopt a Conda-based Python like Miniconda or Anaconda. Conda is a command line tool that allows the creation of separated environments with different python versions and packages. This of it as running multiple virtual machines on the a single host - you can easily switch between any for different needs, or run them simultaneously.
3.5.1 Download and Install

Refer to either of these distributions' page for installation. This guide assumes a conda (Miniconda) installation for setting up Python. If you already have conda, or prefer using the default Python supplied by your OS, skip PyDesigner installation.

3.5.2 Update Conda

First, update conda with

```
$ conda update conda
```

3.5.3 Create new environment

Creating a conda environment is recommended as this will keep all of the dependencies required for this project isolated to just the conda environment called dmri. For more information about conda environments, see The Definitive Guide to Conda Environments. Next, create a conda environment specifically for dMRI preprocessing, called dmri. You can choose any name, but be sure to replace dmri in this guide with any name of your choice.

Next, execute the following two line to create a Python environment ready for PyD installation.

```
$ conda create -n dmri python=3.7
$ conda install -n dmri pip
```

The first line create an environment with Python v3.7, while the second line installs the PyPi package manager.

Once this is all set, you may proceed with the installation of PyD.

3.6 PyDesigner

PyD is an installable Python package designed to perform pre- and post-processing of dMRI acquisitions.

3.6.1 Download

You may clone the main PyDesigner repository for the latest build, or download the build version of your choice from the Releases tab.

To clone the PyDesigner repository, in terminal, run:

```
$ git clone https://github.com/m-ama/PyDesigner.git
```

3.6.2 Install

PyDesigner can be automatically installed with all dependencies by opening a CLI and changing directory to root PyDesigner directory, followed by

```
$ pip install .
```

Note: Remember to switch to your conda environment before parsing this command.
This will execute the `setup.py` script in root directory to automatically configure your Python environment for PyDesigner. When running the automated methods, PyDesigner can simply be called with the command `pydesigner`.

**Note:** If you need a stable, tested and versioned build, download the most recent release from the Release tab. Click on Source code (zip) link and decompress (unzip) to any folder you desire.

## 3.7 NeuroDock

Docker is a container technology designed to package an application and all its needs, such as libraries and other dependencies, into one package. We adapted PyDesigner and its dependencies for compatibility with the Docker Engine to bring DTI/DKI analyses to everyone.

We bring you, **NeuroDock**

NeuroDock is a Docker image containing the most cutting-edge tools required for diffusion and kurtosis imaging. This container was designed for complete dMRI processing pipelines to be platform agnostic. NeuroDock was inspired by the lack of easily-accessible tools across various platforms. NeuroDock is 100% compatible across Windows, Linux, and Mac - while making available the full suite of FSL, MRtrix3 and PyDesigner commands.
3.7.1 Why Docker

By packaging fixed versions of FSL, MRtrix3, and PyDesigner, we are able to guarantee repeatability and consistency across all platforms. Regardless of whether researchers are running Linux, Windows, or Mac OS, identical results can be replicated with Docker technology.

A side-effect to ensuring repeatability with Docker is that it becomes host operating system (OS) agnostic. This allows users to run FSL, MRtrix3, or PyDesigner commands at near-native speed, even on Microsoft Windows.

Additionally, researchers can easily deploy Docker containers to HPCs for rapid processing of large-cohort or longitudinal studies with ease.

3.7.2 Docker vs Virtual Machines

Okay, so you may ask, “why not just load up a VM?”. You have a point. While the two technologies appear to be behaving the same way, at least on the surface level, their inner mechanisms are differ vastly.

Unlike a VM, rather than creating a whole virtual OS loaded with dependencies and other applications, Docker allows applications to share the same OS kernel, thereby providing a significant performance uplift while saving up storage space. With the removal of an entire guest OS in VMs, Docker containers save tons of computational resources that can be diverted towards better performance.

Now that you know some differences, it is time to move on to preparing the Docker image!

3.8 Docker Installation

Docker is relatively straightforward to install and run. Windows and Mac users are able to install Docker like any other GUI-based software package installation. The installation is not dependent on console arguments, like Linux.

Please refer to the instructions below for links and guide.

3.8.1 Linux

Users may refer to the Docker Engine installation guide located here, for installation instructions on their Linux distribution. The steps covered below are targeted for Debian-based or Ubuntu distributions.

Uninstall Docker

1. Uninstall older version or any traces of existing Docker installations

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```

Don’t panic if `apt-get` returns an a warning about missing packages. It’s good they are missing, since we’re trying to purge existing installations of Docker
**Install Docker Engine**

Once all traces of existing Docker installation and dependencies have been purged, you may proceed with the following steps to install the Docker Engine - Community.

1. Update the debian package list with:

```bash
$ sudo apt-get Update
```

2. Install basic packages that enable installation of Docker Engine and its dependencies with:

```bash
$ sudo apt-get install \
apt-transport-https 
ca-certificates 
curl 
gnupg-agent 
software-properties-common
```

3. Add the Docker official GNU Privacy Guard (GPG) key to enable encryption and decryption of communication with the Docker server:

```bash
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

You may verify this key by following the full guide on official Docker documentation, the link to which is located at the beginning of this page.

4. Add the stable Docker Engine repository to your package list with the command:

```bash
$ sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) 
stable"
```

Adding a repository to your Linux distribution allows the OS to pull software packages from the developers’ servers. It directs the OS to the location where these packages are stored.

Then, update your package manager repository with the command:

```bash
$ sudo apt-get update
```

This updates the list of softwares your OS can fetch from various repositories.

5. Once your Debian-based system becomes aware of the Docker Engine, you may install it simply via the command:

```bash
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. Verify your Docker Engine installation with the command:

```bash
$ sudo docker run hello-world
```

If the following information prints in the console window, your Docker Engine installation was sucessful.

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
```
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

https://hub.docker.com/

For more examples and ideas, visit:

https://docs.docker.com/get-started/

You may now proceed with the fetching of NeuroDock Docker image.

## 3.8.2 Mac OS

1. Download Docker Docker Desktop for Mac.

2. Double-click on the downloaded Docker.dmg to start the install process. Follow all on-screen instructions and prompts.

3. Docker should start automatically, indicated by the whale icon in the status bar. Alternatively, you may verify whether Docker is running by parsing the following command in Terminal:

```
$ docker version
```

Or you may run the hello-world container to verify the installation:

```
$ docker run hello-world
```

If you information text being printed into the PowerShell windows, then Docker has been installed successfully.

## 3.8.3 Windows

1. Download Docker Desktop for Windows.

2. Double-click the Docker for Windows Installer to run the installer.

3. Docker should start automatically, indicated by the whale icon in the taskbar. Alternatively, you may verify whether Docker is running by parsing the following command in PowerShell.

```
$ docker version
```

Or you may run the hello-world container to verify the installation:

```
$ docker run hello-world
```

If you information text being printed into the PowerShell windows, then Docker has been installed successfully.
3.9 Docker Configuration

Docker can be configured in a wide-variety of ways based on hardware resources available. Parameters such as CPU cores, RAM and storage can be assigned to Docker for running NeuroDock.

For validations purposes, the NeuroDock image was tested to work as intended on the following three systems:

<table>
<thead>
<tr>
<th>Part</th>
<th>Machine A</th>
<th>Machine B</th>
<th>Machine C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>Apple iMac Pro</td>
<td>Custom</td>
<td>Custom</td>
</tr>
<tr>
<td>OS</td>
<td>Mac OS X Mojave</td>
<td>Ubuntu 18.04</td>
<td>Microsoft Windows 10 Pro</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel Xeon W [8C/16T]</td>
<td>AMD Ryzen R9 2700X [8C/16T]</td>
<td>AMD Ryzen R9 2700X [8C/16T]</td>
</tr>
<tr>
<td>Memory</td>
<td>64 GB</td>
<td>16 GB</td>
<td>16 GB</td>
</tr>
<tr>
<td>Video</td>
<td>Raden Pro Vega 56 8 GB</td>
<td>Nvidia GTX 1080 8 GB</td>
<td>Nvidia GTX 1080 8 GB</td>
</tr>
</tbody>
</table>

We found identical results across the three operating systes on all these configurations.

3.9.1 Docker Preferences

Based on Docker’s system requirements, we recommend assigning the following system resources to Docker:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUs</td>
<td>8</td>
</tr>
<tr>
<td>Memory</td>
<td>16.00 GB</td>
</tr>
<tr>
<td>Disk image size</td>
<td>32.00 GB</td>
</tr>
</tbody>
</table>

By default, Docker assigns itself half the number of available CPU cores and 2 GB of memory. Considering that the entire NeuroDock image is ~14.5 GB, we recommend at least double in disk image size. You may configure your Docker Engine to run on this configuration, or input your own values based on your processing needs. The following sections detail how to set these parameters.

Linux

CPU and memory access to Docker containers on Linux machines is manipulated via CFGS scheduler flags at run time. These flags are:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--cpus=&lt;value&gt;</td>
<td>specify how many CPU cores to use</td>
</tr>
<tr>
<td>-m or --memory</td>
<td>specify the maximum amount of memory available to containers</td>
</tr>
</tbody>
</table>

For a more comprehensive list of manipulable system parameters for for Linux, please visit the Runtime options with Memory, CPUs, and GPUs page on Docker documentation.
**Mac OS**

Manipulating these three variables is very simple on Mac OS because these parameters are located in the GUI.
1. On the Docker icon in the status bar, right-click on the Docker icon, then Preferences.
2. Click on the Resources tab on the left.

![Docker Mac preferences GUI](image)

Fig. 1: Docker Mac preferences GUI; click on resources

3. The Resources menu will show you the configuration, please change them to desired values. You may leave “Swap” at default.

**Windows**

Similar to the Mac, the same sequence of steps apply for the Windows platform.
1. Right-click on the Docker icon in the taskbar, then click on Preferences.
2. Click on the Resources tab on the left.
3. The Resources menu will show you the configuration, please change them to desired values. You may leave “Swap” at default.

Setting the correct configuration will force Docker to not exceed these constraints. By splitting up CPU and memory loads, researchers can process multiple DWIs simultaneously.
3.9. Docker Configuration

Fig. 2: Docker Mac resources configuration

Fig. 3: Docker Windows preferences GUI; configure as desired
3.9.2 GPU Support

At this time, there is no CUDA or ROCm GPU support. These features are planned for a later release. Please use the non-Docker, native Linux configuration to utilize GPU for eddy and EPI correction.

3.10 Install NeuroDock

After successfully installing and configuring Docker, you can install the NeuroDock container in one of two ways:

1. Pulling pre-build image from Docker Hub with `docker pull [image]`
2. Building the image yourself with `docker build [path to image]`

The first option is the recommended method because prebuilt images are guaranteed to work and enhance reproducibility even further. In addition, they are numbered version-controlled for referencing. Your copy of NeuroDock will be configured exactly the same as another person’s.

The second option is intended for developers who make frequent changes to the PyDesigner source code and wish to test their changes in a Docker environment. The Dockerfile script is designed to build a Docker image using PyDesigner in the root directory of the repository.

3.10.1 Docker Hub

Pulling pre-built NeuroDock is incredibly straightforward. Run the following command to pull NeuroDock.

```bash
$ docker pull docker pull dmri/neurodock:tagname
```

where `tagname` is the version you’d like to pull. To install NeuroDock v0.2, you would run the command

```bash
$ docker pull dmri/neurodock:v0.2
```

And that’s it! All you have to do now is to wait for the NeuroDock image to finish downloading.

3.10.2 Local Build

Disclaimer It must be reiterated that this option is reserved for developers; regular users are encouraged to stay away from this method because there is no semantic versioning to reference.

1. Open up a command line interface and change directory to your PyDesigner repository

```bash
$ cd [PyDesigner Repo Path]
```

2. To build a Docker image using your local PyDesigner copy, run the command:

```bash
$ docker build -t [tagname] .
```

Here, `tagname` can be any name you wish to give this image. If you wish to build an image called neurodock, run the command:

```bash
$ docker build -t neurodock .
```

This will build a Docker image called NeuroDock based on your local PyDesigner repository.
3.11 Run NeuroDock

Congratulations, you’ve come this far. You’ve installed Docker and NeuroDock, and are probably wondering how what else to do…

You’re done. Not even kidding! You can now start processing data with PyDesigner and NeuroDock. It’s almost as if FSL, MRtrix3 and PyDesigner commands are built natively into your OS - be it Linux, Mac OS, or even Windows!

3.11.1 Intro to Docker Run

Use the following form of `docker run` command to call all command made available by NeuroDock:

```
$ docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

where,

<table>
<thead>
<tr>
<th>Flag</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>[OPTIONS]</td>
<td>docker options to use when running the container; common options are -it, -v, -d</td>
</tr>
<tr>
<td>IMAGE</td>
<td>image name to run; in this instance, this is dmri/neurodock</td>
</tr>
<tr>
<td>[COMMAND]</td>
<td>specify which NeuroDock commands to run; these can be FSL, MRtrix3 or PyDesigner commands</td>
</tr>
<tr>
<td>[ARG]</td>
<td>arguments for [COMMAND]</td>
</tr>
</tbody>
</table>

Users are encouraged to visit the Docker run reference documentation for more information on controlling the `docker run` command.

3.11.2 Practical Run

The section above covered a generic way to use the `docker run` command. For actual data analysis, we use the following options.

1. `--it --rm` to run docker in interactive TTY mode. What this implies is that your NeuroDock command will run like any other OS commands such as `ipconfig, watch, ls` etc.
2. `--v` to mount the file system or folder to processing

Bind Mount

The second flag, `--v`, makes visible the host’s local filesystem to a Docker container, which otherwise runs in a completely isolated system. By mounting a folder for NeuroDock, you are able to make it process data in said folder. The general guideline is to mount one subject folder at a time. It is advisable that users read through Docker’s bind mounts to understand how Docker containers handle storage.

The correct syntax for the `--v` flag is:

```
--v [HOST PATH TO MOUNT]:[v]
```

Suppose a subject folder `bond_007` in need of processing is structured the following way:

```
bond_007
|-- nifti
```

3.11. Run NeuroDock
This subject needs to be processed PyDesigner read the input nifti files in the nifti directory, and saves the outputs in the processed directory. Since both nifti and processed folders belong to a common parent directory, the bond_007 directory can be mounted to give NeurDock access to both child directories simultaneously.

Here, the directory bond_007 is the [HOST PATH TO MOUNT], the directory that NeuroDock will not be able to see.

Next, we need to define where within the container this directory is mounted, [TARGET AT WHICH TO MOUNT]. You may simply mount this in the root NeuroDock directory at /data.

The flag to reflect this would then be:

```
-v /Users/sid/Desktop/bond_007:/data
```

This would make the contents of host directory bond_007 available in the NeuroDock at /data. Say, for example, the nifti file bond_dwi.nii is located in the host system at /Users/sid/Desktop/bond_007/nifti/bond_dwi.nii. If the above mounting scheme is used, the NeuroDock container will see this file in /data/nifti/bond_dwi.nii.

This filesystem transformation is particularly important when writing scripts for automatic or batch processing of subject directories using the NeuroDock container.

**Put it all together**

Considering everything on this page, it becomes incredibly easy to process a subject using the NeuroDock container. Sticking to bond_007 example above, and combining everything so far, one could process Mr. Bond’s DWI with the command:

```
$ docker run -it --rm -v /Users/sid/Desktop/bond_007:/data \  
  dmri/neurodock pydesigner --standard \  
  --output /data/processed \  
  /data/nifti/bond_dwi.nii, /data/nifti/bond_topup.nii
```

This command runs the --standard PyDesigner pipeline on the input files /Users/sid/Desktop/bond_007/nifti/bond_dwi.nii and /Users/sid/Desktop/bond_007/nifti/bond_topup.nii, and saves all outputs into the directory /Users/sid/Desktop/bond_007/processed
3.12 Dataset Requirements

PyDesigner can process input DWIs in NifTi (.nii), compressed NifTi (.nii.gz), MRTrix3 file format (.mif), and DI-
COM (.dcm) file formats. With the exception of .mif and .dcm filetypes, all other input formats are required to be
accompanied with .bval, .bvec, and .json files.

Note: With the exception of extensions, all files additional accompanying a DWI need to have the same name as DWI. For example, the input DWI file DKI_64_dir.nii will be accompanied by DKI_64_dir.bval,DKI_64_dir.
bvec and DKI_64_dir.json files

3.12.1 Separate or Combined Shells

Having B-value shells in separate or single 4D volumes doesn’t matter as long as each 4D DWI has it's own accom-
pánying files.

3.12.2 JSON File

Every DWI will NEED a .json file of the same name; PyDesigner will refuse to process any input that fails to meet
this criterion. This behavior is intentional to prevent unintentional corrections from being executed when they are
incompatible. Users must create a JSON file if their DICOM to NifTi conversion software fails to create it.

PyDesigner primarily looks for partial Fourier information within a JSON information. This information if encoded
in the fields PartialFourier; or PhaseEncodingSteps and AcquisitionMatrixPE. Users need to have
have these fields at the bare minimum to process DWIs.

3.13 PyDesigner Syntax

PyDesigner has a simple syntax:

```
$ pydesigner [OPTIONS] DWI1,DWI2,DWI3...
```

Mutiple DWIs may be combined by separating their paths with just a comma. For example, one may run PyDesigner
with standard processing using the command:

```
$ pydesigner -s --verbose \
~/Dataset/DKI_B0.nii,~/Dataset/DKI_B1000.nii,~/Dataset/DKI_B2000.nii
```

Simple as that!

3.13.1 DTI, DKI, FBI, or FBWM?

Now that you are ready to process, you may be wondering how to get various metrics from your DWIs. It’s very
simple. PyDesigner figures this out for you. It analyzes input dataset’s B-value shells to extract as much information
as possible. All you, the user, have to do is to just load your DWIs in and grab a drink. Cheers and welcome!
3.14 List of Flags

PyDesigner is extremely flexible when it comes to dMRI processing. Users can easily enable or disable various preprocessing steps without changing the overall sequence.

The list below covers all these flags.

3.14.1 IO Control

These flags allow control of the pipeline’s I/O handling

-o DIR, --output DIR  PyDesigner output directory

3.14.2 Preprocessing Control

Preprocessing control flags allow users to tweak certain parts of the preprocessing pipeline, to accommodate all types of datasets.

-s, --standard  Runs the recommended preprocessing pipeline in order: denoise, degibbs, undistort, brain mask, smooth, rician

-n, --denoise  Denoises input DWI

--extent  Shape of denoising extent matrix, defaults to 5,5,5

--reslice  Reslices input DWI and outputs to a specific resolution in mm or output dimensions

--interp  The interpolation method to use when resizing

-g, --degibbs  Corrects Gibb’s ringing

-u, --undistort  Undistorts image using a suite of EPI distortion correction, eddy current correction, and co-registration. Does not run EPI correction if reverse phase encoding DWI is absent.

--rpe_pairs n  Speeds up topup if a reverse PE is present; specify the number (integer) of reverse PE direction B0 pairs to use

--mask  Computes a brain mask at 0.20 threshold by default

--maskthr  Specify FSL bet fractional intensity threshold for brain masking, defaults to 0.20

--user_mask  Provide path to user-generated brain mask in NifTi (.nii) format

-z, --smooth  Smooths DWI data at a default FWHM of 1.25

--fwhm  Specify the FWHM at which to smooth, defaults to 1.25

-r, --rician  Corrects Rician bias
3.14.3 Diffusion Tensor Control

Users may also tweak computations in estimating DTI or DKI parameters with the following flags.

--nofit             Performs preprocessing only, disables DTI/DKI parameter extraction
--noakc             Disables brute forced kurtosis tensor outlier rejection
--nooutliers        Disables IRLLS outlier detection
--fit_constraints   Specify fitting constraints to use, defaults to 0,1,0
--noqc              Disables saving of quality control (QC) metrics
--median            Performs post processing median filter of final DTI/DKI maps. **WARNING:** Use on a case-by-case basis for bad data only. When applied, the filter alters the values of most voxels, so it should be used with caution and avoided when data quality is otherwise adequate.

3.14.4 Fiber Ball Imaging (FBI) Control

FBI parameters may be fine-tuned with the following flags.

--l_max n           Maximum spherical harmonic degree used in spherical harmonic expansion for fODF calculation

3.14.5 Pipeline Control

These are more general pipeline flags that interface directly with the user or machine.

--nthreads n        Specify number of CPU workers to use in processing, defaults to all physically available workers
--resume            Resumes preprocessing from an aborted or partial previous run
--force             Forces overwrite of existing output files
--verbose           Displays console output
--adv               Disables safety check to force run certain preprocessing steps. **WARNING:** This flag is for advanced users only who fully understand the MRI system and its outputs. Running with this flag could potentially yield inaccuracies in resulting DTI/DKI metrics

3.15 List of Output Files

The number of output files generated by PyDesigner may seem very daunting at first. However, once a certain level of familiarity is achieved, it becomes very easy.

There are generally three types of outputs:

1. **Preprocessing files**: files used in preprocessing; stored in root output directory
2. **Metric files**: DTI/DKI parameters maps, stored in /metrics folder
3. **QC Metrics**: files that enable data quality control; stored in /metrics_qc

The list of ever possible output file is given in the table below.
<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Root Directory</strong></td>
<td></td>
</tr>
<tr>
<td>B0.nii</td>
<td>mean b0 image extracted from processed DWI (exists only if code: --mask is used)</td>
</tr>
<tr>
<td>brain_mask.nii</td>
<td>brain mask extracted from B0.nii (exists only if --mask is used)</td>
</tr>
<tr>
<td>dwi_preprocessed.nii</td>
<td>fully preprocessed DWI Nifti file</td>
</tr>
<tr>
<td>dwi_preprocessed.bval</td>
<td>fully preprocessed DWI's BVAL file in FSL format</td>
</tr>
<tr>
<td>dwi_preprocessed.bvec</td>
<td>fully preprocessed DWI's BVEC file in FSL format</td>
</tr>
<tr>
<td>dwi_preprocessed.json</td>
<td>fully preprocessed DWI's BIDS sidecar</td>
</tr>
<tr>
<td>dwi_raw.nii</td>
<td>raw DWI Nifti file before preprocessing</td>
</tr>
<tr>
<td>dwi_raw.bval</td>
<td>raw DWI's BVAL file in FSL format</td>
</tr>
<tr>
<td>dwi_raw.bvec</td>
<td>raw DWI's BVEC file in FSL format</td>
</tr>
<tr>
<td>dwi_raw.json</td>
<td>raw DWI's BIDS sidecar</td>
</tr>
<tr>
<td>noisemap.nii</td>
<td>noisemap Nifti file produced from denoising (exists only if --denoise is used)</td>
</tr>
<tr>
<td>working.mif</td>
<td>MRtrix3 file formatted DWI that is being preprocessed</td>
</tr>
<tr>
<td>log_command.json</td>
<td>history of preprocessing steps and commands run of DWI</td>
</tr>
<tr>
<td><strong>QC Metrics</strong></td>
<td>root_dir/metrics_qc</td>
</tr>
<tr>
<td>head_motion.png</td>
<td>estimated head motion plotted from displacement field computed by EPI and IRLLS outlier detection</td>
</tr>
<tr>
<td>outliers</td>
<td>snr plots of dwi_raw.nii and dwi_preprocessed.nii</td>
</tr>
<tr>
<td>/fitting/outliers_akc.nii</td>
<td>outliers detected by brute forced kurtosis tensor outlier rejection algorithm</td>
</tr>
<tr>
<td>/fitting/outliers_irlls.nii</td>
<td>outliers voxels detected by irlls outlier detection (4d nifti)</td>
</tr>
<tr>
<td>/eddy</td>
<td>all outputs of the eddy correction (exists if --undistort is used)</td>
</tr>
<tr>
<td><strong>DTI/DKI Metrics</strong></td>
<td>root_dir/metrics</td>
</tr>
<tr>
<td>dti_ad.nii</td>
<td>axial diffusivity map (3d nifti)</td>
</tr>
<tr>
<td>dti_rd.nii</td>
<td>radial diffusivity map (3d nifti)</td>
</tr>
<tr>
<td>dti_md.nii</td>
<td>mean diffusivity map (3d nifti)</td>
</tr>
<tr>
<td>dti_fa.nii</td>
<td>fractional anisotropy map (3d nifti)</td>
</tr>
<tr>
<td>dti_fe.nii</td>
<td>first eigenvalues; represents the principal direction of water (4d nifti)</td>
</tr>
<tr>
<td>dti_trace.nii</td>
<td>sum of diagonals of diffusion tensor (3d nifti); the mean diffusivity (MD)</td>
</tr>
<tr>
<td>dki_ak.nii</td>
<td>axial kurtosis map (3 nifti)</td>
</tr>
<tr>
<td>dki_ri.nii</td>
<td>radial kurtosis map (3d nifti)</td>
</tr>
<tr>
<td>dki_mk.nii</td>
<td>mean kurtosis map (3d nifti)</td>
</tr>
<tr>
<td>dki_kfa.nii</td>
<td>kurtosis fractional anisotropy map (3d nifti)</td>
</tr>
<tr>
<td>dkl_mkt.nii</td>
<td>mean kurtosis tensor (3d nifti); alternative calculation for mean kurtosis</td>
</tr>
<tr>
<td>DT.nii</td>
<td>diffusion tensor (4d nifti; 6 three-dimensional volumes)</td>
</tr>
<tr>
<td>KT.nii</td>
<td>kurtosis tensor (4d nifti; 15 three dimensional volumes)</td>
</tr>
<tr>
<td><strong>FBI/FBWM Metrics</strong></td>
<td>root_dir/metrics</td>
</tr>
<tr>
<td>fbi_zeta.nii</td>
<td>zeta (3d nifti)</td>
</tr>
<tr>
<td>fbi_faa.nii</td>
<td>axonal fractional anisotropy (3d nifti)</td>
</tr>
<tr>
<td>fbi_fodf.nii</td>
<td>FBI FODFs from spherical harmonic expansion (4d nifti)</td>
</tr>
<tr>
<td>fbi_tractography_dsi.fib</td>
<td>FBI FODFs prepared for DSI Studio’s tractography (fib)</td>
</tr>
<tr>
<td>fbwm_awf.nii</td>
<td>axonal water fraction (3d nifti)</td>
</tr>
<tr>
<td>fbwm_Da.nii</td>
<td>intra-axonal diffusivity (3d nifti)</td>
</tr>
<tr>
<td>fbwm_De_ax.ni</td>
<td>axial extra-axonal diffusivity (3d nifti)</td>
</tr>
<tr>
<td>fbwm_De_rad.ni</td>
<td>radial extra-axonal diffusivity (3d nifti)</td>
</tr>
<tr>
<td>fbwm_De_mean.nii</td>
<td>mean radial extra-axonal diffusivity (3d nifti)</td>
</tr>
<tr>
<td>fbwm_fae.ni</td>
<td>extra-axonal fractional anisotropy (3d nifti)</td>
</tr>
<tr>
<td>fbwm_minCost.nii</td>
<td>minimum cost computed by cost function (3d nifti)</td>
</tr>
<tr>
<td>fbwm_minCost.nii</td>
<td>all costs computed by cost function (4d nifti; 100 three dimensional volumes)</td>
</tr>
<tr>
<td><strong>WMTI Metrics</strong></td>
<td>root_dir/metrics</td>
</tr>
</tbody>
</table>

Chapter 3. References
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmti_awf.nii</td>
<td>axonal water fraction (3d nifti)</td>
</tr>
<tr>
<td>wmti_eas_ad.nii</td>
<td>extra-axonal axial diffusivity (3d nifti)</td>
</tr>
<tr>
<td>wmti_eas_rd.nii</td>
<td>extra-axonal radial diffusivity (3d nifti)</td>
</tr>
<tr>
<td>wmti_eas_tort.nii</td>
<td>extra-axonal tortuosity (3d nifti)</td>
</tr>
<tr>
<td>wmti_ias_da.nii</td>
<td>intra-axonal intrinsic diffusivity (3d nifti)</td>
</tr>
</tbody>
</table>

All other files in folder /intermediate_nifti are used by PyDesigner for preprocessing flow control, especially to allow --resume or --force flags to work as intended.

### 3.16 Grants

The PyDesigner project is a result of effort from several contributors and support from grants. This section acknowledges all contributions to the project.

Table 2: List of supporting grants

<table>
<thead>
<tr>
<th>Sponsor</th>
<th>Grant Number</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIH</td>
<td>1R01AG054159</td>
<td>Quantitative Neuroimaging Assessment of White Matter Integrity in the Context of Aging and AD</td>
</tr>
<tr>
<td>NIH</td>
<td>P20GM109040</td>
<td>Institutional Award (IDeA) Center of Biomedical Research Excellence</td>
</tr>
<tr>
<td>NIA</td>
<td>1R01AG057602</td>
<td>Assessing Brain Microstructure in Alzheimer’s Disease with Advanced Diffusion MRI</td>
</tr>
<tr>
<td>NIDCD</td>
<td>R01DC014021</td>
<td>Brain Connectivity Supporting Language Recovery in Aphasia</td>
</tr>
<tr>
<td>NINDS</td>
<td>R01NS110347</td>
<td>Predicting Epilepsy Surgery Outcomes Using Neural Network Architecture</td>
</tr>
</tbody>
</table>

### 3.17 Contributors

Fig. 4: Siddhartha Dhiman, MSc  Research Specialist
The Center for Biomedical Imaging
Department of Neuroscience
Medical University of South Carolina

Fig. 5: Joshua Teves, BSc  Systems Programmer
The Center for Biomedical Imaging
Department of Neuroscience
Medical University of South Carolina

3.16. Grants
Fig. 6: Kayti Keith, BSc
Research Specialist
The Center for Biomedical Imaging
Department of Neuroscience
Medical University of South Carolina

Fig. 7: Benjamin Ades-Aron, MSc
PhD Student
Center for Biomedical Imaging
Department of Radiology
NYU School of Medicine
New York University
Fig. 8: Jens Jensen, PhD
Professor
The Center for Biomedical Imaging
Department of Neuroscience
Medical University of South Carolina

Fig. 9: Els Fieremans, PhD
Assistant Professor
Center for Biomedical Imaging
Department of Radiology
NYU School of Medicine
New York University
Fig. 10: Jelle Veraart, PhD  
Assistant Professor  
Center for Biomedical Imaging  
Department of Radiology  
NYU School of Medicine  
New York University

Fig. 11: Vitria Adisetyo, PhD  
Staff Scientist  
The Center for Biomedical Imaging  
Department of Neuroscience  
Medical University of South Carolina
CHAPTER FOUR

INDICES AND TABLES

- genindex
- modindex
- search