# Configuration & Paths

## Table of contents

💡 Quick reference

Both MATLAB and Python have a single editable path (`project_root` / `PROJECT_ROOT`) that you set once per computer. All other paths are derived automatically.

# 1 Overview

The freely-walking-optomotor system runs across three computers, each with a specific role. A centralised configuration in the code repository ensures all scripts find data in the correct locations.

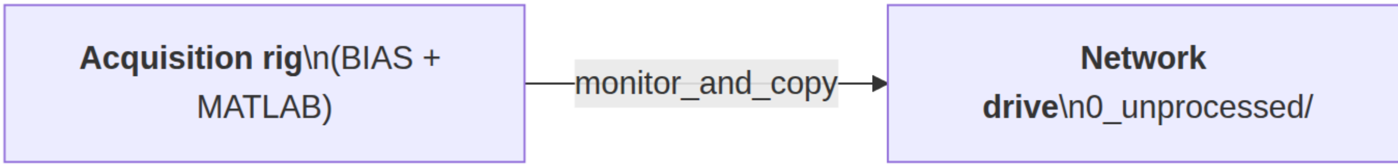
| Computer | Role | Config fields used |
|---|---|---|
| Acquisition rig (Windows) | Runs protocols, records video | `rig_data_folder`, `bias_config`, `SOURCE_ROOT` |
| Processing machine (Windows) | Automated tracking & processing | `project_root` + all local paths + all `NETWORK_*` paths |
| Analysis computer (Mac/any) | Manual analysis, plotting, dashboard | `project_root` + local paths only |

The two config files live in the `config/` directory at the repository root:

- **MATLAB**: `config/get_config.m` — returns a struct via `cfg = get_config()`
- **Python**: `config/config.py` — exports constants via `from config.config import ...`

# 2 Data flow

The automation pipeline moves data through four stages:


Acquisition rig\n(BIAS + MATLAB)
monitor_and_copy
Network drive\n0_unprocessed/

1. **Acquisition rig** records video and LOG files to `rig_data_folder`
2. `monitor_and_copy` transfers completed folders to the network drive (`0_unprocessed/`)
3. `monitor_and_track` copies to local, runs FlyTracker, moves tracked data to `1_tracked/`
4. `daily_processing` runs MATLAB processing, saves results and figures, moves data to `2_processed/`

The processing machine maintains a local copy of the data under `project_root` and synchronises with the network drive. Analysis computers can work from either a local copy or a mounted network drive.

# 3 Setup instructions

## 3.1 Step 1 — Clone the repository

```
git clone https://github.com/leburnett/freely-walking-optomotor.git
```

## 3.2 Step 2 — Create the local data folder

Create the following folder structure (or point to an existing one):

```
your_data_root/
  DATA/
    00_unprocessed/
    01_tracked/
    02_processed/
  results/
  figures/
```

### 3.3 Step 3 — Edit the config files

**MATLAB** — edit `config/get_config.m` (one line):

```
cfg.project_root = '/path/to/your_data_root';
```

**Python** — edit `config/config.py` (one line):

```
PROJECT_ROOT = Path("/path/to/your_data_root")
```

Both files contain commented examples for Windows and Mac.

### 3.4 Step 4 — Add MATLAB paths

Run `setup_path.m` once per MATLAB session (or add it to your `startup.m`):

```
run('/path/to/freely-walking-optomotor/setup_path.m')
```

This adds all `src/` subdirectories to the MATLAB path so that functions like `get_config()`, `process_freely_walking_data()`, and the protocol helper functions are available.

### 3.5 Step 5 — Install the Python environment

```
cd freely-walking-optomotor/python/freely-walking-python
pixi install
```

This installs all Python dependencies (numpy, pandas, scipy, dash, etc.) into an isolated environment managed by pixi.

# 4 Config file reference

## 4.1 MATLAB — `config/get_config.m`

Call `cfg = get_config()` to get the configuration struct.

**Editable:**

| Field | Description |
|---|---|
| `cfg.project_root` | Root of your local data folder — **edit this per computer** |

**Derived from `project_root`** (processing machine + analysis computer):

| Field | Path | Used by |
|---|---|---|
| `cfg.data_unprocessed` | `DATA/00_unprocessed/` | Tracking pipeline |
| `cfg.data_tracked` | `DATA/01_tracked/` | `process_freely_walking_data` |
| `cfg.data_processed` | `DATA/02_processed/` | Archive |
| `cfg.results` | `results/` | Processing, analysis, plotting, dashboard |
| `cfg.figures` | `figures/` | Overview figure output |

**Derived from `repo_root`** (all machines):

| Field | Path | Used by |
|---|---|---|
| `cfg.repo_root` | Auto-detected git repo root | Internal |
| `cfg.patterns` | `src/patterns/Patterns_optomotor/` | Protocol scripts, pattern tools |
| `cfg.calibration_file` | `src/tracking/calibration.mat` | `batch_track_ufmf` |

**Acquisition rig only:**

| Field | Value | Used by |
|---|---|---|
| `cfg.rig_data_folder` | `C:\MatlabRoot\FreeWalkOptomotor\data` | `initialize_video_and_folders` (protocols) |
| `cfg.bias_config` | `C:\MatlabRoot\...\bias_config_ufmf.json` | `initialize_video_and_folders` (BIAS camera) |

**Network drive:**

| Field | Value | Used by |
|---|---|---|
| `cfg.group_drive` | `smb://prfs.hhmi.org/reiserlab/oakey-cokey/data/` | Network access from Mac |

## 4.2 Python — `config/config.py`

Import variables with `from config.config import DATA_TRACKED, RESULTS_PATH`.

**Editable:**

| Variable | Description |
|---|---|
| `PROJECT_ROOT` | Root of your local data folder — **edit this per computer** |

**Derived from `PROJECT_ROOT`** (processing machine + analysis computer):

| Variable | Path | Used by |
|---|---|---|
| `DATA_UNPROCESSED` | `DATA/00_unprocessed/` | `monitor_and_track` |
| `DATA_TRACKED` | `DATA/01_tracked/` | `daily_processing` |
| `DATA_PROCESSED` | `DATA/02_processed/` | `daily_processing` |
| `RESULTS_PATH` | `results/` | `daily_processing`, dashboard |
| `FIGURES_PATH` | `figures/` | `daily_processing` |

**Derived from `REPO_ROOT`:**

| Variable | Path | Used by |
|---|---|---|
| `REPO_ROOT` | Auto-detected repo root | Automation scripts |
| `PATTERNS_DIR` | `src/patterns/Patterns_optomotor/` | Documentation generator |
| `PROTOCOLS_DIR` | `src/protocols/` | Documentation generator |

**Network drive paths** (processing machine only):

| Variable | Path | Used by |
|---|---|---|
| `NETWORK_ROOT` | `\\prfs.hhmi.org\reiserlab\oakey-cokey` | Base for all network paths |

| Variable | Path | Used by |
|---|---|---|
| NETWORK_UNPROCESSED | ...\data\0_unprocessed | monitor_and_copy,<br>monitor_and_track |
| NETWORK_TRACKED | ...\data\1_tracked | monitor_and_track,<br>daily_processing |
| NETWORK_PROCESSED | ...\data\2_processed | daily_processing |
| NETWORK_RESULTS | ...\exp_results | daily_processing |
| NETWORK_FIGS | ...\exp_figures\overview_figs | daily_processing |

**Acquisition rig** (rig computer only):

| Variable | Value | Used by |
|---|---|---|
| SOURCE_ROOT | C:\MatlabRoot\FreeWalkOptomotor\data | monitor_and_copy |

## 5 Network drive structure

The group network drive is the central archive for all experiment data:

```
\\prfs.hhmi.org\reiserlab\oaky-cokey\
  data\
    0_unprocessed/    <- raw data from rig (via monitor_and_copy)
    1_tracked/        <- tracked data (via monitor_and_track)
    2_processed/      <- processed data with MP4 videos (via daily_processing)
  exp_results/        <- .mat result files (via daily_processing)
  exp_figures\
    overview_figs/    <- overview PDF/PNG figures (via daily_processing)
```

**i** UNC vs SMB paths

The Python config uses Windows UNC format (`\\server\share`) while the MATLAB config uses SMB format (`smb://server/share/data/`). The Python `NETWORK_ROOT` points to the share root (`oaky-cokey`); the MATLAB `cfg.group_drive` points to the `data/` subdirectory underneath.

## 6 Local data folder structure

Both the processing machine and analysis computers use the same folder layout under `project_root` / `PROJECT_ROOT`:

```
project_root/
  DATA/
    00_unprocessed/   <- raw data staged for tracking
    01_tracked/       <- tracked data awaiting processing
      YYYY_MM_DD/
        protocol_name/
          strain/
            sex/
              HH_MM_SS/
                LOG_YYYYMMDD_HHMMSS.mat
                REC_.ufmf
                tracking/
                  trx.mat
                  *-feat.mat
    02_processed/     <- fully processed archive
  results/
    protocol_27/
      strain/
        sex/
          YYYYMMDD_HHMMSS_strain_protocol_sex_data.mat
  figures/
    overview_figs/
      ...
```

> ℹ Local vs network numbering
>
> Local folders use zero-padded numbers (`00_`, `01_`, `02_`) while the network drive uses single digits (`0_`, `1_`, `2_`). This is intentional — it makes it easy to tell whether you are looking at a local or network path.

## 7 Automation scripts

The three automation scripts in `python/automation/` run on the processing machine and move data through the pipeline:

| Script | Runs on | Config imports | Purpose |
|---|---|---|---|
| `monitor_and_copy` | Acquisition rig | `SOURCE_ROOT`, `NETWORK_UNPROCESSED` | Watches for completed recordings, copies to network |
| `monitor_and_track` | Processing machine | `DATA_UNPROCESSED`, `DATA_TRACKED`, `NETWORK_*`, `REPO_ROOT` | Copies from network, runs FlyTracker, archives tracked data |
| `daily_processing` | Processing machine | `DATA_TRACKED`, `DATA_PROCESSED`, `NETWORK_*`, `RESULTS_PATH`, `FIGURES_PATH`, `REPO_ROOT` | Runs MATLAB processing, copies results to network |

See the Data Organisation page for details on what each processing stage produces.

# 8 Troubleshooting

**“Path not found” errors in MATLAB** Run `setup_path.m` to add all `src/` subdirectories to the MATLAB path, then check that `get_config().project_root` points to an existing directory:

```
cfg = get_config();
assert(isfolder(cfg.project_root), 'project_root does not exist: %s', cfg.project_root);
```

**Network paths unreachable** The `NETWORK_*` paths require the network drive to be mounted. On Windows this is typically automatic via domain login. On Mac, connect via Finder: Go > Connect to Server > `smb://prfs.hhmi.org/reiserlab/oaky-cokey`.

**Python import errors** Make sure `sys.path.insert` reaches the repo root before importing from `config.config`. The depth depends on the script’s location in the directory tree. For example, scripts in `python/automation/daily_processing/` need four levels:

```
sys.path.insert(0, str(Path(__file__).parent.parent.parent.parent))
```

**Dashboard can’t find data** The dashboard reads from `RESULTS_PATH / "protocol_27"` by default. Ensure you have run `process_freely_walking_data` for the relevant dates, or point the dashboard to a different protocol directory.