Pipeline

Table of contents

This page describes the full experimental workflow — from running an experiment through to processed results — and the automated processing pipeline that handles data flow between machines.

1 Experiment workflow

The data pipeline has five stages:

1. **Run the experiment** — A MATLAB protocol script (e.g. `protocol_27.m`) controls the LED arena and camera, producing a UFMF video file and a `LOG.mat` file with stimulus timing and metadata.
2. **Track the flies** — FlyTracker processes the video offline, extracting each fly's position, heading, and basic features into `trx.mat` (trajectories) and `feat.mat` (features).

3. **Compute behavioural metrics** — `combine_data_one_cohort(feat, trx)` filters bad tracking, interpolates gaps, and computes 12 behavioural metrics (forward velocity, angular velocity, turning rate, distance from centre, etc.).
4. **Split by condition** — `comb_data_one_cohort_cond(LOG, comb_data)` uses LOG frame indices to slice the continuous data into per-condition segments.
5. **Merge experiments** — `comb_data_across_cohorts_cond(protocol_dir)` combines all sessions into a single hierarchical `DATA` struct for group analysis.

Steps **2–3** can be fully automated by the pipeline described below.

2 Automated data pipeline

Three Python scripts (written by Aparna Dev) automate data flow from the acquisition PC to processed results. The scripts run on two physical machines and communicate via shared network directories.

2.1 Overview

The automated pipeline eliminates manual file management between running an experiment and analysing the results. Three scripts, launched via Windows Task Scheduler batch files, handle the progression of data through a series of network directory stages. Each script monitors for new data, processes it, and advances it to the next stage.

Stage	Script	Runs on	Input	Output	Polling
1. Copy	`monitor_and_copy.py`	Acquisition PC	Local experiment data	Network `0_unprocessed`	30 s (watchdog)
2. Track	`monitor_and_track.py`	Processing PC	Network `0_unprocessed`	Network `01_tracked`	5 min, 15 cycles
3. Process	`daily_processing.py`	Processing PC	Local `01_tracked`	Network `2_processed`	One-shot (daily)

2.2 Network directory stages

Directory	Contents	State
`0_unprocessed`	Raw experiment folders (.ufmf + LOG.mat + stamp_log)	Awaiting tracking

Directory	Contents	State
`1_tracked`	Tracked folders (+ trx.mat, feat.mat)	Awaiting processing
`2_processed`	Fully processed folders (+ MP4 videos)	Complete
`exp_results`	Result `.mat` files organised by protocol	Analysis ready
`exp_figures`	Overview figures (PDF/PNG)	QC and review

All network directories reside on the Janelia network share at `\\prfs.hhmi.org\reiserlab\oaky-cokey\data\`.

Completeness check

A folder is considered "complete" (ready to advance to the next stage) when it contains at least one `.ufmf` file, at least one `.mat` file, and a file whose name starts with `stamp_log`. This ensures the experiment has finished writing all outputs before any copying or processing begins.

2.3 Stage 1: Monitor and copy

`monitor_and_copy.py` runs on the **acquisition PC**. It uses the Python `watchdog` library to monitor the data folder for new experiment directories. When a new folder is detected, it is added to a pending set. Every 30 seconds, the script checks pending folders for completeness and copies complete folders to the network `0_unprocessed` directory using `shutil.copytree()`, preserving the full folder hierarchy (`{date}/{protocol}/{strain}/{sex}/{time}/`).

The script runs indefinitely until manually stopped. It is launched on login via `run_monitor_and_copy.bat` registered with Windows Task Scheduler.

2.4 Stage 2: Monitor and track

`monitor_and_track.py` runs on the **processing PC**. It polls the network `0_unprocessed` directory every 5 minutes for new folders. For each untracked folder:

1. Checks whether the folder has already been processed (exists in `1_tracked` or `2_processed`)
2. Copies the folder locally (tracking over the network would be extremely slow)
3. Runs MATLAB FlyTracker in batch mode: `matlab -batch "batch_track_ufmf('<folder>')"`
4. Verifies tracking success by checking for `trx.mat`

5. Moves the tracked data to `1_tracked` on the network and the local archive

The script auto-exits after 15 consecutive scan cycles (75 minutes) with no new data. It is launched on the processing PC at scheduled times via `run_monitor_and_track.bat`.

2.5 Stage 3: Daily processing

`daily_processing.py` runs once daily on the **processing PC**. It is a one-shot script (not a polling loop):

1. Scans `01_tracked` for date folders (`YYYY_MM_DD`) not yet in `02_processed`
2. For each new date, calls: `matlab -batch "process_freely_walking_data('YYYY_MM_DD')"`
3. Copies result `.mat` files to `exp_results` on the network (organised by protocol)
4. Copies overview figures to `exp_figures` on the network
5. Moves the date folder from `01_tracked` to `02_processed` (local and network)
6. Copies generated `.mp4` stimulus videos to `2_processed` on the network

2.6 Supporting scripts

Script	Purpose
`reprocessing_script.py`	Variant of `daily_processing.py` that reprocesses **all** dates (used when processing logic changes)
`copy_movies_to_network.py`	Syncs `.mp4` video files from local `02_processed` to network `2_processed`

2.7 Deployment

Machine	Role	Windows user	Scripts
Acquisition PC	Runs experiments, captures video	`labadmin`	`monitor_and_copy.py`
Processing PC	FlyTracker tracking, MATLAB processing	`burnettl`	`monitor_and_track.py`, `daily_processing.py`

All automation scripts are located in `python/automation/` within the freely-walking-optomotor repository.

3 Pipeline status page

The automation scripts generate a standalone HTML page (`pipeline_status.html`) on the network drive that shows the processing stage of every experiment. It is auto-regenerated whenever the pipeline updates an experiment's status.

3.1 What it contains

The status page includes:

- **Sortable and filterable tables** with colour-coded pipeline stages for each experiment
- **Production experiments** (from September 25, 2024 onwards) — displayed prominently with full metadata. An orange warning indicator flags experiments with missing metadata or missing LOG files.
- **Testing-phase experiments** (before September 25, 2024) — collapsed by default. These older experiments have flat folder structures and "unknown" metadata is expected.
- **Summary charts** — breakdowns by protocol, by strain, and a timeline view (production data only)

3.2 How it is generated

The status page is generated by the `generate_status_page()` function in `python/automation/shared/registry`
It is called automatically whenever the automation pipeline updates an experiment's processing status. The function reads the experiment registry (a JSON file tracking all experiments and their current pipeline stage) and renders the HTML page with the current state.

3.3 How to view it

The status page is saved to the network drive alongside the experiment data:

```
# macOS (requires the network drive to be mounted):
open /Volumes/reiserlab/oaky-cokey/pipeline_status.html

# Windows:
start \\prfs.hhmi.org\reiserlab\oaky-cokey\pipeline_status.html
```