Sci-Agent-STM (MVP)

Windows desktop “visual automation agent” MVP for driving the **Nanonis SPM Control Software** UI using:

- screenshots (full screen + ROIs)
- fixed pixel click anchors
- multimodal agent verification (ROI images)

No NPI, no reverse engineering, no UI element inspection.

Setup

1) Create a venv

```
python -m venv .venv
.\.venv\Scripts\python -m pip install --upgrade pip
.\.venv\Scripts\python -m pip install -r requirements.txt
```

2) Configure UI mappings (workspace)

This MVP assumes the Nanonis window layout/resolution is stable and you calibrate coordinates once.

Option A (recommended): use the GUI calibrator:

```
.\.venv\Scripts\python -m src.calibrate_gui --workspace workspace.json
```

In the calibrator:

- Add/select an ROI or Anchor in the left list
- Click “Draw ROI box” (drag a rectangle) or “Pick anchor point” (single click)
- Click “Save” to write `workspace.json`

Option B: edit JSON directly:

1. Copy `workspace.example.json` → `workspace.json`
2. Edit the ROI rectangles and anchor coordinates:
   - `rois`: screenshot-only regions (readouts, status panels)
   - `anchors`: click targets (input boxes, buttons)

Tip: Use a screen ruler tool or a quick one-off script to print mouse coordinates.

Run

```
.\.venv\Scripts\python -m src.main --agent
```

Agent mode (LLM, multimodal)

Set your OpenAI key in `OPENAI_API` (or `OPENAI_API_KEY`):

```
$env:OPENAI_API = "YOUR_KEY_HERE"
```

Make sure your current Python has the `openai` package installed (recommended: use the repo venv):

```
.\.venv\Scripts\python -m pip install -r requirements.txt
```

Run with `--agent` (uses `gpt-5.2` by default):

```
.\.venv\Scripts\python -m src.main --agent
```

In the TUI, settings are controlled via slash commands:

- `/workspace [path]` (default: `workspace.json`)
- `/model [name]` (get/set)
- `/max_agent_steps [int]` (get/set)
- `/log_dir [path]` (get/set)

Chat commands:

- `/help` (or `/menu`) shows available commands
- `/chat save [name]` saves the current transcript + agent memory to `sessions/`
- `/chat list` lists saved sessions
- `/chat resume <name>` loads a saved session

Scan countdown pause (optional)

If you add an ROI named `scan_time_count_down` (with a clear description of the time format), the agent can use a `pause_scan` tool to read the remaining time from that ROI, wait for the countdown + 5 seconds, then continue. This prevents the agent from spamming observations while a scan is running.

Abort options:

- Move mouse to top-left corner (pyautogui failsafe)
- Press `ESC` (toggle in TUI: `/abort_hotkey on|off`)

Logs

Each run creates a folder under `logs/`:

`logs/<timestamp>/`

- `<step_name>/`
  - `before.png` / `after.png` (when exactly one ROI is captured for the step)
  - `before_<roi>.png` / `after_<roi>.png`
  - `meta.json` (step details)

The system always saves before/after images per step. In agent mode, the model uses ROI images + the last action log to judge whether the UI changed as intended.

Example console output:

```
Plan:
  1. Use `--agent` mode to set a field and click anchors.
Workspace: workspace.json
Dry-run: False
Failsafe: move mouse to top-left to abort (pyautogui.FAILSAFE)
Abort hotkey: ESC
[Agent] Use `set_field`/`click_anchor` actions with anchors from your
workspace.
Logs: logs/20251223_143012
```

Verification

- Rule-based mode: screenshot logging only (no automatic verification)
- Agent mode: multimodal model checks ROI images + action log