Semi-Automated Capture Workflow

1. Overview

This application is a real-time, semi-automated image capture system designed for guided scanning workflows. It utilizes computer vision (OpenCV) to provide the user with real-time feedback on camera stability and movement speed. The system enforces a structured workflow (e.g., Scan Area A $\rightarrow$ Scan Area B) and saves images only when specific stability criteria are met.

2. System Architecture

The application follows a **Model-View-Controller (MVC)** hybrid pattern, heavily relying on **multi-threading** to ensure the UI remains responsive while computationally expensive computer vision tasks run in the background.

Core Components

- **AssetManager:** Handles loading of static resources (images, icons) and cross-platform audio playback.
- **GuidanceSystem (Model/Logic):** The core processing engine. It runs on a background thread, calculating optical flow to determine camera stability and speed.
- **SessionManager (Controller):** Manages the workflow state (e.g., which anatomical area is being scanned), handles file I/O, and processes user inputs.
- **OverlayDrawer (View):** Responsible for rendering the User Interface (UI), including status bars, instructions, and buttons, directly onto the video feed.

3. Class Breakdown

3.1. `AssetManager`

- **Role:** Central repository for external assets.
- **Key Functionality:**
  - Loads icons (`Lower.png`, `Upper.png`) with error handling.
  - `play_voice(filename)`: Abstraction layer for playing audio cues.

3.2. `GuidanceSystem`

- **Role:** Real-time computer vision analysis.
- **Threading:** Runs a daemon thread (`_motion_worker`) to process frames without blocking the UI.
- **Algorithm (Optical Flow):**

    1. **Region of Interest (ROI):** Crops the center of the frame to focus on the subject.
    2. **Feature Detection:** Uses `cv2.goodFeaturesToTrack` to find trackable points.
    3. **Lucas-Kanade Flow:** Uses `cv2.calcOpticalFlowPyrLK` to track points between frames.
    4. **Metrics:** Calculates the **Mean (μ)** (Speed) and **Standard Deviation (σ)** (Stability) of the movement vectors.
- **Hysteresis:** Implements a smoothing filter (`HysteresisState`) to prevent the UI warnings from flickering rapidly when near threshold values.

3.3. `SessionManager`

- **Role:** State machine and file management.
- **States (`ScanningState`):**
    1. `READY_TO_SCAN_LOWER`
    2. `SCANNING_LOWER` (Active processing)
    3. `READY_TO_SCAN_UPPER`
    4. `SCANNING_UPPER` (Active processing)
    5. `COMPLETE`
- **File I/O:** Automatically names files with timestamps (`LOWER_{timestamp}.jpg`) and saves them to a local `Captures` directory. Maintains a list of session files to allow for "Recapture" (undo) functionality.

3.4. `OverlayDrawer`

- **Role:** UI Rendering.
- **Features:**
    - Draws the "Traffic Light" status bar (Green=Ready, Cyan=Arming, Amber/Red=Warning).
    - Overlays transparent PNG icons onto the frame.
    - Draws interactive buttons (Main Action and Recapture).
    - Handles the "Flash" effect logic upon successful image capture.

4. Logic & Algorithms

4.1. The Capture Logic (Stability Check)

The system does not simply take pictures on a timer. It uses a "stable-trigger" mechanism:

1. **Motion Analysis:** Checks if Speed (μ) < `CAPTURE_SPEED_THRESH` and Stability (σ) < `CAPTURE_STAB_THRESH`.
2. **Arming:** If stable, a timer starts (`stable_since`).
3. **Trigger:** If stable for `CAPTURE_DELAY_S` (0.5s), and the cooldown from the previous shot has passed, a capture is triggered automatically.

4.2. Visual Guidance Colors

- **Green:** Stability is good.
- **Cyan:** Stability is good, "Arming" capture (Hold position).
- **Amber:** User is moving slightly too fast or shaking.
- **Red:** User is moving significantly too fast.

5. Configuration

Key parameters can be adjusted in the `GuidanceSystem` `__init__` method:

- `MOTION_TARGET_WIDTH`: Resolution for optical flow processing (lower = faster performance).
- `CAPTURE_SPEED_THRESH`: Sensitivity to movement speed.
- `CAPTURE_STAB_THRESH`: Sensitivity to shakiness.
- `CAPTURE_COOLDOWN_S`: Minimum time between two photos.