Objectives

1. Resolve the direct stitch issues, including memory management inefficiencies and performance bottlenecks in line handling.
2. Transition from a vector-based system to a double 2D pointer system for improved performance and memory efficiency.
3. Address challenges in sequencing between neighbor value removal and direct stitch operations.
4. Refactor pointer-related sub-functions into a dedicated file for better modularity and maintainability.
5. Add new functionality for resetting and clearing image states to improve user control.

Activities

1. **Direct Stitch Issues Resolution:**
   - Investigated and resolved issues with memory management, coordinate handling, and stitching accuracy.
   - Replaced the vector-based storage system with a double 2D pointer (`DarkLine**`) system for storing and accessing line data.
   - Implemented custom memory management routines, including initialization, cleanup, validation, and capacity tracking.
   - Enhanced line detection accuracy and added robust error-handling mechanisms for stitching operations.
   - Improved state preservation during operations with deep copying techniques.
2. **Conversion to Double 2D Pointer System:**
   - Transitioned from a vector-based storage approach to a pointer-based array system for line data management.
   - Created functions for memory allocation, deallocation, and safety checks to manage resources efficiently.
   - Reduced memory fragmentation, improved processing speed, and enhanced CPU utilization by eliminating vector overhead.
3. **Addressed Sequencing Challenges:**

    - Found that removing lines via neighbor values before direct stitch operations caused errors.
    - Resolved the issue by introducing dedicated workflows for neighbor value and direct stitch removals.
    - Added state preservation with the `saveCurrentState()` function and implemented deep copies to avoid conflicts.
4. **Code Refactoring:**
    - Separated pointer-related sub-functions into a new header and source file named `pointer_operations`.
    - Functions refactored and modularized included:
        - `handleNeighborValuesRemoval`, `handleDirectStitchRemoval`, `generateRemovalSummary`, etc.
5. **New Features Implementation:**
    - Added `resetToOriginal()` to revert image changes.
    - Added `clearImage()` to reset the image state.
    - Connected `clearAllDetectionResults()` to reset detection lines.

Achievements

1. Successfully resolved direct stitch issues, improving memory management, processing speed, and stitching accuracy.
2. Completed the transition to a double 2D pointer system, significantly enhancing performance and resource efficiency.
3. Developed and implemented robust workflows for managing neighbor value and direct stitch operations.
4. Refactored pointer-related functions into a dedicated file, improving code modularity and maintainability.
5. Enhanced user functionality with new reset and clear features for better control over image processing.

Problems & Solutions

1. **Problem:** Direct stitch operations failed due to memory leaks and invalid dimensions in the DarkLineArray.
   Solution: Introduced a double pointer system with proper memory management routines, including validation and cleanup.
2. **Problem:** Sequence conflicts between neighbor value removal and direct stitch operations caused incomplete processing.
   Solution: Developed separate handlers and workflows for each operation, preserving states and implementing proper sequencing.
3. **Problem:** Vector-based approach caused performance bottlenecks and inefficient memory usage.
   Solution: Replaced vectors with a double 2D pointer system, enhancing data access patterns and reducing CPU overhead.
4. **Problem:** Overly large setup functions in the control panel were difficult to manage.
   Solution: Moved pointer-related sub-functions to `pointer_operations` files, modularizing and simplifying the code structure.