📝 Image Processing Report

📌 Objective

This script performs multi-phase analysis and transformation on an image loaded from a CSV file. The goal is to visualize the image, identify specific patterns (such as facial features), modify those features, apply noise reduction while preserving key pixels, and summarize findings.

🔍 Phase 1: Loading and Revealing the Image

- **Image Source**: A grayscale image loaded from `/content/secret_image .csv`.
- **Initial Display**: The image is visualized using four different colormaps:
  - Grayscale (`gray`)
  - Heatmap (`hot`)
  - Cool tones (`cool`)
  - Perceptually uniform (`viridis`)
- **Output**: Visualization is saved as `phase1_visualizations.png`.

□ Phase 2: Pattern Detection and Analysis

- **Black Pixel Detection**: Identifies pixels with a value of 0 (black).
  - **Count**: Number of black pixels found.
  - **Coordinates**: All coordinates of detected black pixels are listed.
- **Bounding Box Calculation**:
  - Computes a rectangular boundary around the black pixels.
  - **Output**: Displayed with a red rectangle and saved as `phase2_bounding_box.png`.
- **Pattern Recognition**:
  - **Mouth Detection**: Searches for rows with multiple consecutive black pixels.
  - **Eye Detection**: Looks for horizontally symmetric pairs of black pixels at the same vertical position.
  - **Face-like Structure**: Declared if both eyes and mouth patterns are detected.

🎨 Phase 3: Modifying the Image

- **Grayscale to RGB**: The grayscale image is normalized and converted to an RGB format.
- **Feature Editing**:
  - **Eyes**: Detected clusters are marked red.
  - **Mouth**: Flipped vertically to create a sad face expression.
- **Blue Border Addition**:

- A border of 5-pixel width is added around the image in blue.
- **Output**: Modified image is saved as `phase3_modified_image.png`.

□ Phase 4: Noise Reduction (Mean Filter)

- **Protected Pixels**:
    - Specific pixels are protected from filtering (black pixels, red eye clusters, and specified coordinates).
- **Mean Filter**:
    - A `uniform_filter` of size 2 is applied to reduce noise.
    - Only non-protected pixels are altered.
- **Comparison Visualization**:
    - Side-by-side display of noisy vs. denoised image.
    - **Output**: Saved as `phase4_denoised_comparisonWithoutEyes.png`.

❓ Phase 5: Summary and Answers

1. **Black Pixels Found**:
   → `{num_black_pixels}` black pixels detected.
2. **Coordinates of Black Pixels**:
   → First 5: `{black_pixel_coordinates[:5]} ...` and more.
3. **Bounding Box**:
   → If found: `(min_y, min_x, max_y, max_x) = {bounding_box}.`
4. **Detected Features**:
    - If structure found: "Face-like pattern detected."
        - Eyes: Positions of top 2 detected eye pairs.
        - Mouth: Rows where mouth pattern was found.
    - If no structure: “No structured features detected.”

📂 Generated Output Files

File Name	Description
`phase1_visualizations.png`	Displays original image in 4 color maps.
`phase2_bounding_box.png`	Shows black pixel region with bounding box.
`phase3_modified_image.png`	Modified image with red eyes, sad mouth, blue border.
`phase4_denoised_comparisonWithoutEyes.png`	Comparison of noisy vs. denoised image.

✅ Final Remarks

This code demonstrates structured image analysis by combining **NumPy**, **Matplotlib**, and **SciPy** techniques. It effectively identifies features, applies transformations, and preserves critical visual data during noise reduction—useful for tasks like **image pattern recognition**, **filtering**, and **simple computer vision experiments**.