UI Component Comparison Tool - Project Report

Project Overview

Goal: Compare two UI screenshots (base and test) to detect missing, misplaced, and overlapped components. Calculate similarity using both structural and visual metrics. Annotate differences visually for debugging and regression testing.

Key Functionalities

- Image Preprocessing
- Component Detection
- Component Comparison
- Annotation
- Similarity Computation

Algorithmic Pipeline

1. Preprocessing: Grayscale, blur, edge detection (Canny), dilation, and morphological closing.
2. Component Detection: Contour extraction with size filtering.
3. Component Comparison:
   a. Matching: Centroid proximity.
   b. Misplacement: Reverse matching from test to base.
   c. Overlapping: IOU > threshold.
4. Drawing Annotations: Using OpenCV rectangles and labels.
5. Similarity Metrics: Component match score and SSIM.

Justification of Algorithms

Preprocessing: Effective on UI screenshots without training data.

Component Detection: Fast, template-free.

Comparison: Centroid + IOU provides a balance of precision and speed.

SSIM: More aligned with human perception than MSE or PSNR.

Visualization: Intuitive debugging via bounding boxes.

UI Component Comparison Tool - Project Report

Pros and Cons

Pros:

- No need for labeled data
- Fast and interpretable
- Scalable

Cons:

- Sensitive to large layout shifts
- Struggles with complex visual variations

Alternatives Considered

1. Deep Learning (YOLO/SSD): Accurate but requires labeled datasets.
2. Template Matching: Not robust to layout changes.
3. Pixel-wise diff or MSE: Not perceptually meaningful.

Why This Approach

Balances performance and simplicity for UI regression testing. Effective without needing ML training.

Potential Enhancements

- OCR integration
- Deep learning-based component detection
- Interactive GUI
- Dynamic thresholding based on layout