Proposal for Google Summer of Code 2025

Image output evaluation for testing of print/scan job processing

Google
Summer of Code

Open Printing

making printing just work

Personal Information:

- **Name:** Sanskar Yaduka
- **Email:** sanksaryaduka4444@gmail.com
- **GitHub:** Sanskary2303
- **Time Zone:** IST (UTC + 5:30)
- **University:** Indian Institute of Technology, Kanpur
- **Major:** Mechanical Engineering (Bachelor of Technology)

Project Overview:

The objective of this project is to develop an automated system for evaluating the quality of print and scan jobs. Using OpenCV for graphical analysis and OCR for text verification, the system will assess output quality based on parameters like content preservation, glyph detection, border alignment, and resolution consistency. This will aid in ensuring that OpenPrinting's processing pipeline maintains high fidelity in image reproduction and text clarity.

Background and Motivation:

OpenPrinting's current testing methodology only detects processing errors, crashes, infinite loops, or empty outputs, but doesn't verify content accuracy. This leaves a critical gap in quality assurance, as visually correct output is essential for a printing system. My project aims to close this gap by implementing intelligent image comparison techniques that can determine if a human would perceive the output as matching the input, with appropriate tolerance for acceptable variations.

Project Details

Problem Statement

Print/scan job processing requires verification that goes beyond simple binary pass/fail metrics. The challenges include:

- Text may render differently but remain readable
- Colors may have slight variations but remain visually similar

- Document layout may shift slightly but maintain structural integrity
- Processing artifacts may be introduced but not affect overall usability

Proposed Solution

I'll implement a modular framework that leverages computer vision libraries (OpenCV), OCR tools (Tesseract), and structured comparison methodologies to evaluate output quality across multiple dimensions:

1. **Text Accuracy:** Ensure all text is preserved and readable using OCR comparison
2. **Structural Integrity**: Verify document layout and structure using feature detection
3. **Color Fidelity**: Confirm colors are reproduced accurately within acceptable tolerances
4. **Edge Quality**: Check for clarity of lines and edges in graphics
5. **Overall Visual Quality**: Compute comprehensive quality metrics

Technical Implementation

The implementation will consist of:

1. **Core Comparison Engine:**
    - Image preprocessing pipeline for normalization
    - Multiple comparison strategies (pixel-based, feature-based, text-based)
    - Configurable tolerance thresholds for different document types
2. **Integration Modules:**
    - Interface with existing OpenPrinting test frameworks
    - Input/output handlers for various document formats (PDF, images, text)
    - Compatibility with diffoscope for detailed differences reporting
3. **Reporting System:**
    - Visual difference highlighting and annotation
    - Detailed metrics reporting with actionable insights
    - Integration with CI pipelines for automated quality gates
4. **Test Document Generator:**
    - Creation of diverse test documents (text, graphics, mixed content)
    - Edge case generators for stress testing

Integration with Existing Projects

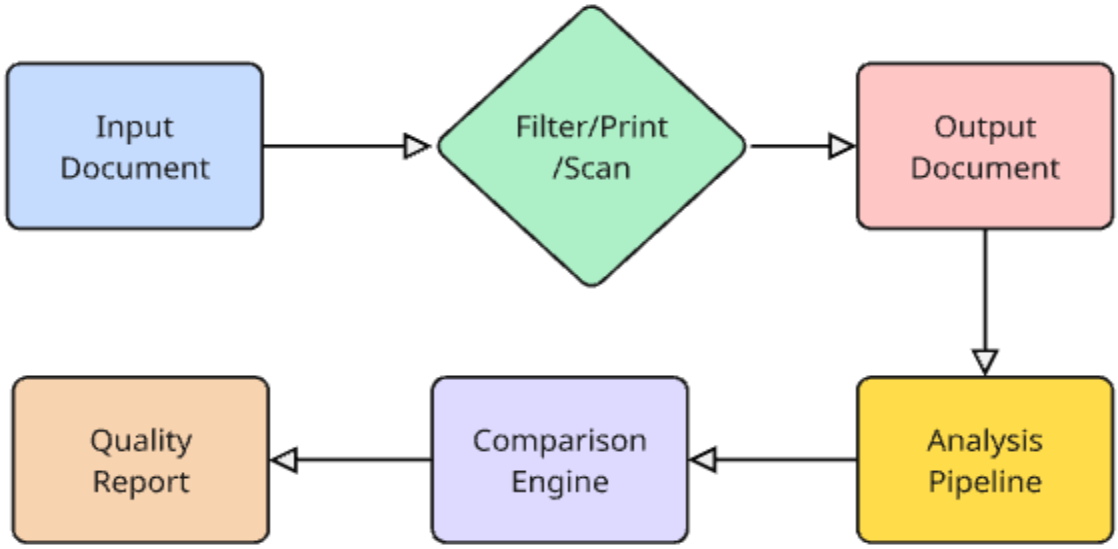
This work will directly benefit:

- **libcupsfilters CI testing**: Enhanced verification of filter output quality
- **MFP simulation**: Validation of simulated printer output against expected results
- **Fuzz-based protocol testing**: Better evaluation of fuzzing test outputs

---

Technical Approach

Architecture



Key Technologies

- **OpenCV**: For image processing and feature detection
- **Tesseract OCR**: For text extraction and comparison
- **diffoscope**: For detailed file comparisons
- **C**: For efficient implementation of core components
- **Python**: For scripting, test generation, and reporting

Technical Details and Algorithms

1. **Image Comparison Metrics:**
    - Structural Similarity Index (SSIM): Measures perceived changes in structural information
    - Peak Signal to Noise Ratio (PSNR): Evaluates level of noise and distortion
    - Mean Squared Error (MSE): Quantifies average squared error between pixels
    - Histogram comparison: Evaluates color distribution similarity

2. **Feature Detection and Matching:**
    - SIFT/ORB feature detection to identify keypoints in documents
    - Feature matching to verify structural integrity between input/output
    - Homography calculation to account for slight rotations or scaling

3. **Text Comparison Implementation:**

```
def compare_text_content(original_img, processed_img):
    # Extract text using Tesseract OCR
    original_text = pytesseract.image_to_string(original_img)
    processed_text = pytesseract.image_to_string(processed_img)
    # Calculate similarity using sequence matching
    similarity = SequenceMatcher(None, original_text,
processed_text).ratio()


    # Identify missing/added content
    missing_content = set(original_text.split()) -
set(processed_text.split())
```

```
    added_content = set(processed_text.split()) -
set(original_text.split())

    return similarity, missing_content, added_content
```

4. **Edge Quality Analysis:**

```
def edge_quality_comparison(original_img, processed_img):

    # Extract edges using Canny edge detector

    original_edges = cv2.Canny(original_img, 100, 200)

    processed_edges = cv2.Canny(processed_img, 100, 200)



    # Calculate edge preservation metrics

    edge_similarity = cv2.matchShapes(original_edges,
processed_edges, 1, 0.0)



    return edge_similarity
```

Potential Challenges and Mitigation Strategies

1. **Complex Document Layouts Challenge**: Documents with complex layouts, tables, or mixed content may be difficult to compare accurately.

   Mitigation: Implement region-based analysis that treats different document areas with specialized comparison algorithms. For example, detect text regions, image regions, and graphical elements separately.

2. **Performance Optimization Challenge**: Image comparison operations can be computationally expensive, especially for high-resolution documents.

Mitigation: Implement progressive resolution analysis, starting with low-resolution comparison for quick assessment and escalating to higher resolution only when needed. Utilize parallel processing for independent comparison metrics.

3. **Threshold Calibration Challenge**: Finding optimal tolerance thresholds that distinguish between acceptable variations and actual content problems.

   Mitigation: Create a calibration system that analyzes known good/bad document pairs to automatically determine optimal thresholds. Implement adaptive thresholds based on document type and content complexity.

4. **Edge Cases and Corner Cases Challenge**: Handling documents with extreme properties like very low contrast, unusual fonts, or heavy graphics.

   Mitigation: Build a comprehensive test suite with edge case documents. Implement fallback comparison strategies when primary methods fail, with clear reporting of confidence levels.

Project Timeline:

Community Bonding Period (4 weeks)

- Engage with OpenPrinting mentors and the community.
- Study the existing CUPS and driverless printing infrastructure.
- Set up the development environment and integrate OpenCV, Tesseract, and diffoscope.
- Finalize architecture and implementation plan.

Phase 1: Core Framework Development (4 weeks)

Week 1-2: Implement Basic Image Comparison Framework

- Develop an image preprocessing pipeline (grayscale conversion, normalization, noise reduction).
- Implement pixel-based comparison techniques (MSE, PSNR, SSIM) for detecting differences.
- Develop a script to automate comparisons of sample print outputs.

Week 3-4: Develop OCR-Based Text Comparison Module

- Integrate Tesseract OCR to extract text from scanned images.
- Implement a text similarity checker using Levenshtein distance and SequenceMatcher.
- Develop an accuracy metric to quantify text content discrepancies.
- Deliverable: A functional text and pixel-based comparison module.

Phase 2: Enhanced Analysis Features (4 weeks)

Week 5-6: Implement Feature Detection and Structural Comparison

- Use SIFT/ORB to detect key features in document layouts.
- Implement homography estimation to handle minor transformations (rotation, scaling).
- Validate the structural integrity of print output using keypoint matching.

Week 7-8: Develop Color and Edge Quality Evaluation

- Implement histogram comparison techniques to measure color fidelity.
- Develop an edge preservation metric using Canny edge detection.
- Optimize image processing pipeline for efficiency.
- Deliverable: A multi-dimensional document analysis pipeline.

Phase 3: Integration and Testing (4 weeks)

Week 9-10: CI Integration and Automated Testing

- Integrate the framework with OpenPrinting's CI system (libcupsfilters testing suite).
- Develop automated scripts to validate print/scan quality at different stages.
- Test the system with diverse print/scan outputs.

Week 11-12: Develop Comprehensive Test Suite and Documentation

- Generate synthetic test cases for different document types (text, graphics, mixed content).
- Implement edge case tests (low contrast, rotated scans, partial prints).
- Write detailed documentation for framework usage and contribution guidelines.
- Deliverable: Fully functional tool integrated into OpenPrinting's testing ecosystem.

Final Week: Project Wrap-Up

- Buffer for unexpected issues and fine-tuning.
- Code cleanup and documentation refinement.
- Prepare a final report and presentation for the OpenPrinting community.
- Deliverable: A stable, well-documented, and community-tested print/scan evaluation framework.

Expected Outcomes:

- A fully functional print/scan evaluation tool.
- Automated testing framework to validate CUPS and driverless printing workflows.
- Comprehensive documentation and open-source contribution to OpenPrinting.

About Me:

I am a third-year undergraduate student at IIT Kanpur, pursuing a B.Tech in Mechanical Engineering, with a strong technical background in systems programming, cybersecurity, robotics, and software development. My expertise spans low-level system design, penetration testing, drone swarm intelligence, and full-stack development.

I have worked extensively with **C, C++, Python and Bash**, along with frameworks and tools like **ROS, MAVLink, FastAPI, Flask, Django, and Flutter**. My passion for system-level programming led me to develop **PyShell**, a custom Unix-like shell in Python, and **MyGit**, a lightweight version control system built from scratch to replicate core Git functionalities.

In the domain of **cybersecurity**, I have conducted vulnerability assessments of IIT Kanpur's infrastructure at C3i, identifying critical OWASP Top 10 vulnerabilities. I also built a custom vulnerability scanning tool for automated security testing.

I have also worked on **computer vision** and GUI development. My experience includes OpenCV-based image processing, OCR-based text recognition, and real-time object tracking. As part of my Human Tracking Drone project, I developed a real-time human detection and tracking system using MobileNet SSD for object recognition and MiDaS for depth estimation, enabling drones to autonomously follow and track targets. This experience, combined with my expertise in **low-level programming and security**, makes me well-suited for this project.

Why Me?

- I have **prior experience contributing to open-source projects** and have **created a GitHub repository** demonstrating my initial work on this project: https://github.com/Sanskary2303/OpenPrinting-Image-Evaluation
- I am proactive in engaging with the OpenPrinting community and have already **started familiarizing myself with CUPS and driverless printing**.
- My strong background in **computer vision and system-level programming** aligns well with the project's objectives.

Contributions So Far

I have already made some progress by:

1. Creating a document pipeline testing framework
2. Implementing various image comparison metrics
3. Developing a test document generator
4. Building HTML report generation capabilities
5. Creating cleanup utilities to manage test files

Post-GSoC Plans

I'm committed to maintaining and extending this project beyond GSoC. Specifically:

- Continuing to improve the analysis algorithms
- Supporting integration with other OpenPrinting projects
- Contributing to documentation and user guides
- Helping with bug fixes and feature requests

Communication

I will:

- Provide weekly updates on project progress
- Attend all scheduled meetings with mentors
- Be available daily on preferred communication platform by the mentor

- Maintain clear documentation throughout development
- Commit code regularly with descriptive commit messages

Conclusion:

The successful implementation of this project will significantly enhance OpenPrinting's quality assurance capabilities by providing automated, intelligent verification of print and scan output. By leveraging computer vision and OCR technologies, we can ensure that printed documents appear as expected to human users, increasing the reliability and quality of open source printing solutions.