Indian Institute of Information Technology (IIIT) Nagpur

Department of CSE

Torn-Image Reconstruction using Robust Registration, Homography, and Seam-Aware Blending

Project Report

N. Krithin

ID: BT23CSE131

Shravan

ID: BT23CSE103

A. Aditya

ID: BT23CSE114

Vasu

ID: BT23CSE095

Dhruv

ID: BT23CSE136

Course Name: DIP Minor / Image Processing Project

Semester: 5 *Branch:* CSE

Under the guidance of

Dr. Tapan Jain

Nagpur - 441108 (India)

Date: 13 November, 2025

Abstract

We present a practical system that reconstructs a single image from two torn parts. The pipeline is robust to arbitrary tear orientation, rotations (0/90/180/270 degrees), flips/mirroring, and partial inversions. It combines: (1) exhaustive orientation search with edge-compatibility scoring, (2) feature-based registration using SIFT/ORB and RANSAC-estimated homography, and (3) seam-aware blending with post-processing for border removal and sharpening. A lightweight SQLite database logs runs, metadata, and outputs to support repeatable experiments. Experiments on printed pages, notebook photos, and synthetic tears demonstrate reliable reassembly with clean seams across varied conditions.

Keywords

Image registration; torn image; homography; SIFT; ORB; RANSAC; blending; OpenCV; SQLite.

1. Introduction

Reconstructing documents or photographs from torn pieces is important in document forensics, archival restoration, and day-to-day recovery of damaged notes. Torn halves form a single planar surface with a dominant seam and often minimal overlap. Challenges include unknown orientation (rotations/inversions), uneven lighting, perspective distortion from phone capture, and low-texture regions. This project builds a deterministic, reproducible pipeline that accepts two image parts, searches orientations, aligns parts with robust feature matching and homography, falls back to edge-based alignment if features are scarce, blends seams, removes borders, and logs results in an SQLite database.

2. Literature Review

Key classical components relevant to this project:

- **Scale-Invariant Features:** SIFT (Lowe, 2004) — robust keypoint detection and descriptors.
- **ORB:** A fast binary descriptor alternative (Rublee et al., 2011).
- **Robust Estimation:** RANSAC for outlier rejection when estimating homographies (Fischler & Bolles, 1981).
- **Approximate Nearest Neighbors:** FLANN for efficient descriptor matching (Muja & Lowe, 2009).
- **Seam/Exposure Handling:** Multiresolution/Laplacian blending and graph-cut seam selection (Burt & Adelson, 1983).
- **Edge detection:** Canny edge detector for structural cues (Canny, 1986).
- **Local contrast:** CLAHE for better feature extraction in low-contrast regions.

Most image stitching systems expect overlapping photos from different viewpoints; torn-piece reconstruction differs because pieces are complementary and orientation unknown. Our contribution is an exhaustive orientation-and-edge search that guides homography and seam-aware fallback blending for robust reconstruction.

3. Problem Statement and Objectives

Given two images that originated from a single planar image torn into two parts, reconstruct the original despite:

- unknown rotations (0/90/180/270) and flips,
- perspective distortions and illumination differences,
- low-feature or repetitive textures,
- arbitrary tear directions.

Objectives:

- Robust orientation discovery via edge-compatibility scoring.
- Use feature-based registration and RANSAC homography when possible.
- Provide deterministic fallback using edge-based alignment and seam-aware blending.
- Deliver a command-line tool with reproducible logging to SQLite.
- Provide quantitative and qualitative evaluation metrics.

4. System Model

The pipeline has three high-level blocks: preprocessing & orientation search, registration (homography or fallback), and seam handling + post-processing. A short textual flow: Load images → Preprocess (grayscale + CLAHE) → Exhaustive orientation/edge search → Best configuration → Feature detection/matching → If enough inliers: estimate homography + warp → Else: edge-based alignment → Blend seam → Remove borders and enhance → Save output and log to DB.

4.1 High-Level Pipeline

1. **Input & validation:** read two images and basic checks (size, colorspace).
2. **Preprocessing:** CLAHE-enhanced grayscale for stable features; color retained for final composite.
3. **Orientation search:** exhaustive rotations and flips for both images; score candidate edge pairings using combined metrics.
4. **Registration:** if sufficient matches found, estimate RANSAC homography and warp the second piece; otherwise fall back to edge-based alignment.
5. **Seam handling:** seam-aware blending (nonlinear feather), border crop, and

light sharpening.

6. **Persistence:** save output and insert run metadata into an SQLite database for reproducibility.

4.2 Module Breakdown

- **Loader:** path checks and OpenCV imread wrapper.
- **Preprocessor:** CLAHE on grayscale, optional gamma or color normalization.
- **Orientation & Edge Scorer:** rotations $\in \{0, 90, 180, 270\}$ and flips $\in \{\text{none}, \text{h}, \text{v}, \text{hv}\}$. Edges considered: top/bottom/left/right; compatibility score combines normalized cross-correlation, pixel similarity, and Canny-edge similarity on narrow edge strips.
- **Feature Detector/Matcher:** SIFT (preferred), fallback ORB; FLANN or BFMatcher as appropriate; Lowe's ratio test applied.
- **Model Estimator:** cv2.findHomography(..., RANSAC) with tunable reprojection threshold. Compute canvas bounds and warp the secondary piece.
- **Fallback Stitcher:** size normalization across seam direction and nonlinear feather blending across a narrow band (alpha ramp with exponent).
- **Post-Processing:** black-border removal, bilateral denoising, mild unsharp masking.
- **Database:** simple SQLite schema to log runs and metrics.

5. Algorithmic Details

5.1 Orientation Search

For each image generate 4 rotations $\times$ 4 flips. For each transformed pair, extract narrow strips at compatible edges (e.g., top vs bottom) and compute:

$$\text{score} = 0.4 \cdot \text{NCC} + 0.4 \cdot \text{pixel-sim} + 0.2 \cdot \text{Canny-sim}$$

The best configuration and confidence are selected for downstream processing.

5.2 Feature Matching and Homography

- Detect keypoints/descriptors on CLAHE-enhanced grayscale images.
- KNN match ($k = 2$) and apply the ratio test $d_1 < r \cdot d_2$ with $r \approx 0.7{-}0.75$.
- Estimate homography using RANSAC; accept if a) H found, b) inlier count ≥ 10, and c) inlier ratio ≥ 0.3 (tunable).
- Warp the second image into the first image's frame and composite by overlaying with mask and blend.

5.3 Edge-Based Fallback

If feature-based homography is unreliable, use the best edge pairing:

- Normalize size along seam direction.
- Define overlap band (up to 100 px) and apply a nonlinear alpha ramp ($\alpha^{0.5}$ style) across the band.
- Blend and crop.

5.4 Post-Processing

- Trim black borders by thresholding and taking the largest contour bounding box.
- Denoise with a bilateral filter and apply a mild high-pass sharpening kernel; clamp pixel values to [0,255].

6. Implementation Details

6.1 Languages and Libraries

- Python 3.9+
- OpenCV (cv2), NumPy, SQLite3, itertools
- Optional: `opencv-contrib-python` to enable SIFT

6.2 Project Structure (suggested)

```
src/improved_reconstructor.py
data/input/part1.jpg
data/input/part2.jpg
outputs/reconstructed.jpg
db/image_reconstruction.db
docs/report.pdf
slides/presentation.pptx
```

6.3 How to Run

```
# 1) Install dependencies
pip install opencv-python opencv-contrib-python numpy

# 2) Run
python3 improved_reconstructor.py path/to/part1.jpg path/to/part2.jpg outputs/result
```

Console output reports keypoints, matches, homography success/fallback, confidence, and output path. SQLite DB is created/updated automatically.

6.4 Database Schema

- **reconstructions**: id (PK), image1_path, image2_path, output_path, created_at, status, method, rotation_detected
- **reconstruction_metadata**: id (PK), reconstruction_id (FK), keypoints_found, matches_found, confidence

7. Dataset and Experiment Design

7.1 Data

- **Real:** photos of torn notebook pages, printed A4 sheets, and photos of physically torn pieces.
- **Synthetic:** split a single image into two halves, randomly rotate/flip and apply small perspective warps to simulate capture conditions.

7.2 Protocol

For each pair:

1. Run the tool.
2. Record keypoints, matches, inliers, chosen orientation, edge pair, confidence score, and method used.
3. Save the output and crop boundaries.

7.3 Evaluation Metrics

- Inlier ratio: inliers / total matches.
- SSIM/PSNR over a band around the seam (when ground truth is available).
- Edge visibility score: gradient magnitude difference across seam vs neighboring areas.
- User study: 1–5 rating of seam visibility and alignment accuracy.
- Runtime and memory usage.

8. Results

Include the following in the report (place images in the folder and reference them here):

- Input pair examples (varied rotations/inversions).
- Matched keypoints visualization with inlier mask.
- Final reconstruction with crop box overlay.
- A table of metrics per test case (inliers, SSIM near seam, runtime).

9. Discussion

Strengths: Works when either homography or edge cues dominate; orientation search automates rotation/flip discovery; lightweight and reproducible (no deep models required).
Weaknesses: Very low-texture or highly damaged seams may drift; lighting/shadows can leave faint seams; feather blending may not fully hide large exposure mismatches.

10. Conclusions

We presented an end-to-end, orientation-aware pipeline for reconstructing torn images. Combining exhaustive orientation search with robust feature-based homography and a seam-aware fallback yields reliable reconstructions across varied captured conditions. The system is fast to run, easy to deploy, and logs experiments for reproducibility.

11. Future Work

- Graph-cut or dynamic programming seam optimization and Poisson blending for near-invisible seams.
- Exposure compensation and color balancing modules.
- Learned orientation/edge scoring (small CNN) to improve confidence estimation.
- Sub-pixel seam alignment using elastic refinement along the tear curve.
- Multi-piece reconstruction (pairwise compatibility graph and MST assembly).
- GUI for interactive seam correction.

12. Risk, Ethics, and Limitations

Risk: false matches leading to misalignment; mitigate via stricter inlier thresholds and sanity checks.
Privacy/Ethics: avoid storing sensitive images long-term; the DB stores paths and metrics only.
Limitations: assumes a planar original; large missing regions or occlusions reduce performance.

13. Project Management

Milestones (suggested):

1. Week 1–2: Data collection and baseline stitching; DB schema.
2. Week 3–4: Orientation search + scoring; unit tests.
3. Week 5: Homography and fallback integration; logging.
4. Week 6: Blending/polish and evaluation scripts.

5. Week 7: Report, slides and demo video.

14. How the Code Works (Step-by-Step)

1. `load_images()`: validate paths and read BGR images.
2. `find_best_configuration()`: exhaustive rotations $\times$ flips for both images and all edge pairs; compute best orientation and confidence.
3. `detect_features_robust()`: CLAHE $\rightarrow$ SIFT (fallback ORB).
4. `match_features_robust()`: FLANN/BFMatcher KNN with ratio test.
5. `stitch_with_homography()`: if ≥ 4 good matches and RANSAC finds H, warp image2 into image1 frame.
6. `stitch_with_alignment()`: if homography fails, choose seam direction and perform nonlinear feather blending.
7. `remove_black_borders()`: threshold and crop to largest non-background contour.
8. `enhance_result()`: bilateral filter + light sharpening.
9. `save_to_database()`: insert run metadata into SQLite and save final image.

15. Experiment Logging (DB)

Insert runs into `reconstructions` with input paths, output path, status (SUCCESS/-FAIL), method used (HOMOGRAPHY/FALLBACK), and detected rotation. Insert metrics into `reconstruction_metadata` (keypoints, matches, inliers, confidence).

16. Presentation Outline

1. Problem demo: two randomly rotated torn pieces.
2. Pipeline overview diagram.
3. Orientation search explanation with edge-strip visualizations.
4. Feature matching visual with inliers and homography.
5. Fallback stitching and blending demonstration.
6. Quantitative table (inliers, SSIM near seam, runtime).
7. Live demo: run CLI on new pieces.
8. Conclusion and future work.

17. Appendix

A. Environment

- Python 3.9+
- pip packages: `opencv-python`, `opencv-contrib-python`, `numpy`
- Tested on Windows and Linux; CPU-only.

B. CLI Examples

```
python3 improved_reconstructor.py data/input/partA.jpg data/input/partB.jpg outputs/
python3 improved_reconstructor.py scans/left.png scans/right.png
```

C. Troubleshooting

- If SIFT not available: install `opencv-contrib-python` or rely on ORB fallback.
- If outputs are skewed: verify both inputs belong to the same original and reduce motion blur.
- If dark borders remain: increase threshold in `remove_black_borders()`.

D. Pseudocode (high level)

```
# high-level reconstruct flow
img1, img2 = load_images(p1, p2)
best_cfg, confidence = find_best_configuration(img1, img2)
img1_t, img2_t = apply_transformations(img1, img2, best_cfg)
kps1, desc1 = detect_features(img1_t)
kps2, desc2 = detect_features(img2_t)
matches = match_features(desc1, desc2)
H, inliers = estimate_homography(matches, kps1, kps2)
if H is valid and inliers >= threshold:
    warped = warp(img2_t, H)
    result = blend_with_mask(img1_t, warped)
else:
    result = edge_based_stitch(img1_t, img2_t, best_edge)
result = remove_borders_and_enhance(result)
save(result); log_to_db(...)
```

Listing 1: high_level_pseudocode