PrintChakra

Complete Processing Pipeline & Technologies

Document Processing System v2.1.0

Generated: 2025-10-25 09:25:04

Table of Contents

1. Executive Summary

PrintChakra is a modular document processing system using computer vision, AI-powered OCR, and real-time communication. It automates document capture, enhancement, recognition, and conversion through a sophisticated 12-stage pipeline.

Core Features:
- 12-stage sequential processing pipeline
- Multi-method document detection with geometric scoring
- Advanced image enhancement with CLAHE and histogram equalization
- Tesseract OCR with 15-attempt optimization (3 configs x 4 preprocessing variants)
- Document classification using KNN with 8 feature extraction methods
- Real-time progress updates via WebSocket (Socket.IO)
- Batch and single-file processing with comprehensive error tracking

2. Technology Stack

Backend Framework:
- Flask 3.0 - REST API server
- Flask-SocketIO 5.3.5 - WebSocket real-time communication
- Python 3.8+ - Core language

Computer Vision & Image Processing:
- OpenCV 4.10 - Edge detection, contour analysis, perspective transforms
- NumPy - Array operations, mathematical computations
- Pillow - Image format conversion and manipulation

OCR & Text Recognition:

- Tesseract OCR - Multi-engine text extraction (Legacy + LSTM)
- PyTesseract - Python wrapper for Tesseract

Machine Learning:

- scikit-learn - KNN classifier for document type prediction
- Feature extraction from image statistics and geometry

File Conversion & Export:

- img2pdf - High-quality image-to-PDF conversion
- PyMuPDF (fitz) - PDF manipulation and image extraction
- python-docx - Word document generation

Frontend & Communication:

- React 19 - UI framework
- TypeScript 4.9.5 - Type-safe development
- Chakra UI 2.10.3 - Component library
- Socket.IO client 4.8.1 - WebSocket communication

Deployment & Automation:

- Vercel - Frontend deployment
- ngrok - Secure tunneling for remote access
- PowerShell - Deployment automation scripts

3. Processing Pipeline Architecture

Stage 1: Quality Validation

Measures image blur and focus metrics.

Stage 2: Color Space Conversion

BGR to grayscale: gray = 0.299*R + 0.587*G + 0.114*B

Stage 3: Threshold & Binarization

Adaptive thresholding (block size: 11x11)

Stage 4: Noise Removal

Non-Local Means Denoising (NLMeans)

Stage 5: Edge Detection

Canny edge detection (thresholds: 50, 150)

Stage 6: Contour Detection

External contour extraction from edge map

Stage 7: Perspective Correction

4-point homography transform for straightening

Stage 8: Contrast Enhancement

Brightness boost + histogram eq + CLAHE

Stage 9: Morphological Operations

Erosion and dilation (3x3 kernel)

Stage 10: OCR Processing

Tesseract OCR (3 PSM x 4 variants = 12 attempts)

Stage 11: Image Optimization

JPEG compression (quality: 90)

Stage 12: File Storage & Export

Save image, text, metadata, optional PDF

4. Mathematical Framework & Scoring

4.1 Contour Scoring Function

Total_Score = margin_score + rect_score + aspect_score + area_score + solidity_score

Components:
- margin_score: Penalty if contour touches image boundary
  (Threshold: margin < 4% = -600 penalty)
- rect_score: Measures corner angles closeness to 90 degrees
  (< 8 degrees error = +100)
- aspect_score: Prefers document aspect ratios 1.2-2.5
  (e.g., A4 = 1.414)
- area_score: Area ratio 10-70% of image (+100), >80% = -400
- solidity_score: Convexity measure (solidity > 96% = +50)

Acceptance: Total_Score > 50

4.2 Geometric Coordinates & Transforms

Corner Ordering by angle from center:
theta = arctan2(y - center_y, x - center_x)

Perspective Transform (Homography):
[x']   [h11 h12 h13] [x]
[y'] = [h21 h22 h23] [y]
[w']   [h31 h32 h33] [1]

Corner Refinement (Inset):
refined = corner + (center - corner) / |center - corner| * inset_pixels
Standard inset: 12-15 pixels to avoid shadow boundaries

Normalized Coordinates (0-100):
x_norm = (x / image_width) * 100

4.3 Margin Analysis

For detected contour corners:
left_margin = min(x) / width
right_margin = (width - max(x)) / width
top_margin = min(y) / height
bottom_margin = (height - max(y)) / height
min_margin = minimum of all four margins

Scoring:
- min_margin < 0.04: score = -600 (background edge)
- 0.04-0.06: score = -300
- 0.06-0.12: score = -50
- >= 0.12: score = +100 (good centering)

5. Document Detection Algorithm

5.1 Multi-Method Approach

Method 1 - Canny Edge Detection:
- Gaussian blur: 7x7 kernel
- Canny thresholds: (45,125), (55,160), (70,200) - three levels
- Morphological ops: dilate (5x5, 2 iter), erode (2x2, 1 iter)
- Area filters: > 8000 px2, < 75% image_area

Method 2 - Adaptive Thresholding:
- Gaussian kernel, block size 17x17
- Morphological closing: 7x7 kernel, 2 iterations
- Polygon approximation: epsilon = 0.020-0.038 * perimeter

5.2 Corner Refinement

Refined corner avoids shadow boundaries:
P_orig = original corner (x,y)
C = mean of all corners (center)
d = (C - P_orig) / ||C - P_orig|| (unit direction)
P_refined = P_orig + d * inset_pixels

Standard inset: 12 pixels for typical shadows

6. Image Enhancement Techniques

6.1 Multi-Stage Contrast Enhancement

Stage 1 - Brightness Boost:
I_bright = I_gray + 25

Stage 2 - Histogram Equalization (Blended):
I_eq_full = equalize(I_bright)
I_eq = (1-0.4) * I_bright + 0.4 * I_eq_full

Stage 3 - CLAHE:
I_clahe = CLAHE(I_eq, clipLimit=2.0, tileGridSize=8x8)

Stage 4 - Final Blend:
I_enhanced = 0.5 * I_eq + 0.5 * I_clahe

Result: Improved text visibility with preserved texture

6.2 OCR Preprocessing Variants

Variant 1: Bilateral filter (diameter=9, sigma=75,75)
Variant 2: Adaptive threshold (blockSize=11, C=2)
Variant 3: Adaptive threshold (blockSize=15, C=3)
Variant 4: CLAHE + sharpening
Variant 5: High contrast (CLAHE clip=3.0)

4 variants x 3 OCR configs = 12 OCR attempts
Best result selected by text length

7. OCR Multi-Configuration System

7.1 Tesseract Configurations

PSM 3 (Automatic): Auto layout detection
PSM 4 (Column): Single column of text
PSM 6 (Block): Uniform text block

OEM 3: Default (Legacy + LSTM if available)

Multi-Config Strategy:
- 4 preprocessing variants x 3 PSM modes = 12 attempts
- Selection: Result with maximum text_length
- Output: Best text, character/word/line count, confidence

7.2 Confidence Scoring

Per-word confidence (0-100 scale):
OCR_Confidence = mean(confidence values where conf > 30)

High-confidence threshold: conf > 60%
Metrics: char_count, word_count, line_count, avg_confidence

8. Classification & Feature Extraction

8.1 KNN Classifier (8 Features)

Feature 1: Aspect Ratio = width / height
Feature 2: Text Density = non-zero pixels / total
Feature 3: Edge Density = edge pixels / total
Feature 4: Horizontal Lines (angle < 10 or > 170 degrees)
Feature 5: Vertical Lines (80 < angle < 100 degrees)
Feature 6: Mean Intensity (brightness)
Feature 7: Intensity Std Dev (contrast)
Feature 8: Contour Complexity (number of regions)

Classes: ID, BILL, RECEIPT, FORM, NOTE, OTHER

KNN: k=3 neighbors, Euclidean distance

8.2 Feature Normalization

X_normalized = (X - mean) / stddev

Ensures equal feature weight and improves classification

9. Coordinate Transformations & Perspective

9.1 Four-Point Transform (Homography)

Input: 4 source points (top-left, top-right, bottom-right, bottom-left)

Calculate output dimensions:
width_A = distance(bottom-left to bottom-right)
width_B = distance(top-left to top-right)
max_width = max(width_A, width_B)

height_A = distance(top-right to bottom-right)
height_B = distance(top-left to bottom-left)
max_height = max(height_A, height_B)

Output points: rectangle (0,0), (max_width,0), (max_width,max_height), (0,max_height)

H = getPerspectiveTransform(source_points, output_points)
I_corrected = warpPerspective(I_input, H, (max_width, max_height))

9.2 Distance Metrics

Euclidean distance: ||v|| = sqrt(x2 + y2)

Used for corner ordering, margins, and refinement calculations

10. Real-time Orchestration & Progress

10.1 Socket.IO Event Flow

Backend -> Frontend Events:
- processing_progress: {step, total_steps, stage_name, message}
- processing_complete: {filename, text, stats, duration}
- processing_error: {error, stage_failed, filename}

Frontend listens and updates:
- Status indicator (green/red connection)
- Progress bar (step / total_steps * 100%)
- Stage name and real-time messages
- File list with status badges

10.2 Batch Statistics

Tracked metrics:
- total_files: Number of files
- successful: Completed count
- failed: Failed count
- success_rate = (successful / total_files) * 100%
- errors: List of error messages

Output: Detailed batch report with breakdown

10.3 Transport Layer

Primary: WebSocket (WSS for HTTPS) - Low latency
Fallback: HTTP Long-Polling - Firewall compatible

Configuration:
- reconnectionAttempts: 10
- reconnectionDelay: 1000ms initial
- reconnectionDelayMax: 5000ms maximum
- timeout: 15000ms

Works through ngrok and HTTPS endpoints

Conclusion

PrintChakra combines advanced computer vision, mathematical optimization, and real-time communication to deliver a robust document processing system. The multi-stage pipeline with geometric scoring, multi-config OCR, and real-time orchestration ensures high-quality document digitization.