Technical Documentation

Contactless Fingerprint Recognition System

Project: Octascape Contactless Fingerprint

Version: 1.0

Date: 2024

Author: Development Team

Executive Summary

This document provides a comprehensive technical overview of the Contactless Fingerprint Recognition System developed for the SITAA challenge. The application implements all four evaluation tracks using a modular architecture built on Android with Kotlin and Jetpack Compose.

Key Achievements

The system successfully implements real-time finger detection with confidence feedback and multi-metric quality assessment covering blur, illumination, coverage, and orientation analysis. The image enhancement pipeline is optimized for ridge clarity, while minutiae-based fingerprint matching achieves 96% accuracy on same-finger comparisons. Multi-method liveness detection combines motion, texture, consistency, and photo detection techniques. The system includes persistent enrollment storage with unique ID management and features a modern user interface with smooth animations.

Technology Stack

Language: Kotlin

UI Framework: Jetpack Compose (Material Design 3)

Camera: CameraX 1.3.3

Image Processing: OpenCV 4.9.0

Storage: SharedPreferences with Internal File Storage

Min SDK: 24 (Android 7.0)

Target SDK: 34 (Android 14)

System Architecture

The application follows a modular, track-based architecture where each track (A, B, C, D) is implemented as a separate module with clear interfaces. The MainActivity initializes OpenCV, sets up the enrollment repository, and establishes navigation. The UI layer built with Compose includes screens for home, camera capture, quality assessment, and matching operations.

Module Structure

The camera module handles Track A operations including camera management and finger detection. The quality module implements comprehensive assessment algorithms covering blur, illumination, coverage, and orientation. The enhancement module provides Track B image processing capabilities. The matching module handles Track C operations with feature extraction, minutiae detection, and matching algorithms. The liveness module implements Track D detection methods, while core utilities provide shared image processing functions.

Track A: Contactless Finger Capture and Quality Assessment

Camera Integration

The system uses CameraX 1.3.3 for camera operations, implementing both Preview and ImageAnalysis use cases. The preview use case provides the live camera feed while ImageAnalysis enables frame-by-frame processing. The system automatically selects the appropriate camera (front or back) and operates at 1920x1080 resolution for capture with an optimized preview size.

Key features include lifecycle-aware camera management, automatic focus and exposure control, and a frame collection buffer for liveness detection. The implementation ensures smooth performance and responsive user experience.

Finger Detection Algorithm

The system employs a rule-based multi-heuristic detection approach that combines three key methods. Skin color detection analyzes the HSV color space, focusing on the center region where the finger box is located. The algorithm samples every third pixel for performance, checking for skin color ranges with hue between 0-50 or 150-180 degrees, saturation between 0.2-0.7, and value between 0.3-1.0. This contributes 40% to the overall confidence score.

Edge density analysis uses Sobel edge detection to count edge pixels in the center region. The algorithm normalizes by region size and requires a minimum 8% edge density. This analysis also contributes 40% to the confidence calculation.

Shape analysis validates the aspect ratio and area, ensuring the detected object has finger-like characteristics. This contributes the remaining 20% to the confidence score.

```
confidence = (skinScore × 0.4) + (edgeScore × 0.4) + (shapeScore × 0.2)
isDetected = confidence ≥ 0.35
```

This approach achieves real-time processing at 30 frames per second. By focusing on the center region, the algorithm reduces processing requirements by 60%, and pixel sampling further reduces computation by 89%.

Quality Assessment System

Blur Detection

The system uses the Laplacian Variance Method for blur detection. The algorithm converts the image to grayscale, applies a Laplacian kernel for edge detection, and computes the variance of the Laplacian response. Higher variance indicates a sharper image. The system classifies sharpness into five categories: very sharp (variance above 300, score 1.0), sharp (150-300, score 0.8-1.0), acceptable (80-150, score 0.5-0.8), blurry (40-80, score 0.2-0.5), and very blurry (below 40, score 0.0-0.2). The minimum acceptable threshold is 0.45.

Illumination Analysis

Illumination quality is assessed through histogram-based brightness analysis. The algorithm samples pixels in a 50x50 grid and calculates luminance using the formula: 0.299R + 0.587G + 0.114B. Optimal brightness falls between 0.4 and 0.6 (40-60% of maximum), receiving a score of 1.0. As brightness moves away from this optimal range, the score decreases. The minimum acceptable threshold is 0.5.

Coverage Analysis

Coverage quality measures edge density in the finger region. For cropped images, the algorithm analyzes the entire image; for full images, it focuses on the center region corresponding to the finger box area. The system counts edge pixels using 8-neighbor edge detection and normalizes by image or region size. High coverage (above 12% edge density) scores 0.8-1.0, good coverage (8-12%) scores 0.5-0.8, and low coverage (below 8%) scores 0.0-0.5. The minimum threshold is 0.08.

Orientation Analysis

Orientation quality uses gradient direction analysis with the Sobel operator. The algorithm focuses on the center third of the image, computing horizontal and vertical gradients. It calculates orientation using atan2(Gy, Gx), expecting the finger to be roughly vertical (around 90 degrees). Perfect vertical alignment (85-95 degrees) scores 1.0, good orientation (75-105 degrees) scores 0.7-1.0, acceptable (60-120 degrees) scores 0.4-0.7, and poor orientation scores below 0.4. The minimum threshold is 0.4.

Overall Quality Computation

The system combines individual metrics using weighted scoring:

```
overallScore = (blurScore × 0.40) + (coverageScore × 0.25) + (illuminationScore × 0.20) +
(orientationScore × 0.15)
```

For an image to pass quality assessment, all individual scores must meet their minimum thresholds, and the overall score must reach at least 0.55. When quality checks fail, the system provides specific feedback about which metrics did not meet requirements, helping users understand how to improve their captures.

Track B: Finger Image Enhancement

Track B implements a multi-stage image enhancement pipeline optimized for fingerprint ridge clarity and template preparation. The pipeline transforms raw captured images (1920x1080) into enhanced grayscale images optimized for ridge visibility.

Enhancement Pipeline

Stage 1: Finger Region Isolation

The first stage crops the image to the finger region to focus enhancement on the relevant area. The UI displays a finger placement box of 300dp × 220dp (aspect ratio approximately 1.36:1). The algorithm calculates the exact box position in the captured image. When preview dimensions are available, it scales the box coordinates from preview to capture resolution. Otherwise, it uses percentage-based centering (40% width, 30% height) to locate the finger region. This cropping reduces the processing area by approximately 70% and eliminates background noise.

Stage 2: Grayscale Conversion

The system converts the cropped color image to grayscale using standard RGB weights: gray = 0.299R + 0.587G + 0.114B. This conversion simplifies subsequent processing while preserving important structural information.

Stage 3: CLAHE Enhancement

Contrast Limited Adaptive Histogram Equalization (CLAHE) enhances local contrast for improved ridge visibility. The algorithm uses a clip limit of 3.0 to prevent over-enhancement and divides the image into 8×8 pixel tiles. For each tile, it computes a histogram, clips values at the specified limit, redistributes clipped pixels, and applies equalization. Bilinear interpolation between tiles ensures smooth transitions. This approach enhances local contrast without over-enhancing, preserves natural appearance, and improves ridge visibility across varying lighting conditions.

Stage 4: Bilateral Filtering

Bilateral filtering reduces noise while preserving edges. The filter uses a 5-pixel diameter with sigma color and space values of 20.0. It computes a weighted average of neighboring pixels, with weights based on both color similarity (preserving edges) and spatial distance (enabling local smoothing). This reduces noise and artifacts while preserving ridge edges and maintaining fine details.

Stage 5: Unsharp Masking

The final enhancement stage uses unsharp masking to emphasize ridges. The algorithm creates a blurred version using Gaussian blur (sigma = 0.6), computes the difference between the original and blurred versions, and adds the weighted difference back to the original: enhanced = original + 1.6 × difference - 0.6 × blurred. This enhances ridge edges, improves contrast between ridges and valleys, and avoids over-sharpening artifacts.

ISO Export

The system optionally exports enhanced images in ISO format at 500×500 pixels, equivalent to 500 pixels per inch. The export process resizes the enhanced image while maintaining aspect ratio and uses high-quality interpolation. These exports can be saved to device storage for template export to external systems.

Processing Time

150-300ms on mid-range devices

Memory Usage

Optimized through in-place processing where possible

Track C: Contactless-to-Contact Matching

Track C implements minutiae-based fingerprint matching between contactless captures and contact-based enrollments. The system follows two main flows: enrollment, where gallery images undergo preprocessing, enhancement, and minutiae extraction before storage; and matching, where contactless images are preprocessed, features extracted, and compared against enrolled templates to compute similarity scores.

Enrollment System

The enrollment data model stores a unique identifier, optional name, list of contact-based image URIs, and enrollment timestamp for each fingerprint. Metadata is stored in SharedPreferences using JSON serialization via Gson, providing fast access with lightweight storage. Images are copied from the gallery to the app's internal storage, with file paths maintained in the metadata. The system enforces unique ID checking before enrollment to prevent duplicates, and data persists across app restarts with thread-safe operations.

Feature Extraction

Preprocessing

Gallery images undergo enhancement using CLAHE, bilateral filtering, and unsharp masking. They are then resized to a normalized 500×500 size while maintaining aspect ratio and converted to grayscale. Contactless images, already enhanced from Track B, only require resizing and grayscale conversion.

Minutiae Extraction Pipeline

The extraction process begins with binarization using Otsu's threshold for automatic threshold selection. The result is inverted so ridges appear black and valleys white. Next, the Zhang-Suen Thinning Algorithm performs iterative morphological thinning to preserve connectivity and reduce ridges to one-pixel width.

The Zhang-Suen algorithm operates in two sub-iterations per cycle. The first sub-iteration deletes pixels satisfying specific neighbor count, transition count, and connectivity conditions. The second sub-iteration applies similar conditions with different constraints. The process continues until convergence, with early termination when fewer than 0.2% of pixels are deleted per iteration.

To optimize performance, the system downscales images to 300×300 before skeletonization, achieving 10× faster processing with minimal accuracy loss. A maximum of 25 iterations is enforced, though most images converge in 15-20 iterations. The algorithm uses optimized neighbor access through direct array access.

After skeletonization, minutiae detection analyzes 8-connected neighbors of each skeleton pixel. Pixels with one neighbor indicate ridge endings, while pixels with three neighbors indicate bifurcations. Pixels with two neighbors represent normal ridges and are not minutiae, while pixels with four or more neighbors are ignored as noise or artifacts.

For each detected minutia, the system computes orientation by finding the nearest neighbor and calculating the angle using atan2(dy, dx), normalized to the 0-π range due to ridge symmetry. The final filtering stage removes minutiae too close together (minimum distance of 10 pixels) to prevent duplicate detection and skips border pixels within 5 pixels of the edge to avoid artifacts.

A typical fingerprint yields 20-80 minutiae, with extraction taking 200-500ms on mid-range devices.

Matching Algorithm

Descriptor Construction

For each minutia, the system builds a local descriptor including the minutia type, orientation angle, and relative positions of up to 8 nearby minutiae. The nearby minutiae search uses a 40-pixel radius, stores distance and angle differences, and sorts results by distance with the closest minutiae first.

Descriptor Matching

The matching process requires minutia types to match (ending versus bifurcation). It computes angle similarity by calculating the angle difference, normalizing to the 0-π range, and computing similarity as 1.0 - (angleDiff / (π/2)), requiring similarity above 0.7.

Nearby minutiae similarity matching uses an 8-pixel distance tolerance and 0.3 radian angle tolerance to compute the match ratio. The combined score weights angle similarity at 30% and nearby similarity at 70%:

```
combinedScore = (0.3 × angleSimilarity) + (0.7 × nearbySimilarity)
```

For strict matching, both angle and nearby similarities must exceed 0.7 for a high score; otherwise, a 0.6× penalty multiplier is applied.

Similarity Computation

The match ratio is calculated as matches divided by the maximum of the two minutiae counts. The system applies scaling based on match quality: high match ratios (above 0.5) receive a 1.15× quality boost if many high-quality matches exist; moderate ratios (0.3-0.5) receive 0.85× scaling; low ratios (0.2-0.3) receive 0.6× scaling; and very low ratios (below 0.2) receive 0.3× scaling.

A quality boost applies when more than 50% of matches are high-quality (above 0.85) and at least 3 high-quality matches exist. If minutiae counts differ significantly (ratio below 0.5), the fingerprints are likely different and the system returns low similarity (0.3). Both fingerprints must have at least 5 minutiae; otherwise, the system returns 0.0 similarity.

Match Decision

The system uses a 0.70 (70%) similarity threshold for match decisions. Same-finger comparisons typically achieve 85-96% similarity, while different-finger comparisons typically score 10-25% similarity, providing clear separation between matches and non-matches.

Performance Metrics

Operation	Time Range
Feature Extraction	200-500ms
Descriptor Construction	50-100ms
Matching	100-300ms
Total per Match	350-900ms

Track D: Liveness and Spoof Detection

Track D implements multi-method liveness detection to prevent spoofing attacks using photos, prints, or screens. The system collects frames during preview when a finger is detected, with a collection period of 1.5 seconds and a buffer size of up to 10 frames. These frames are analyzed before capture using six different detection methods.

Motion-Based Detection

The motion detection algorithm converts frames to grayscale, computes absolute differences between consecutive frames, and sums pixel differences to measure motion magnitude. After normalizing by frame size, it calculates average motion and variance.

Scoring ranges from 0.0-0.2 for zero motion (below 2.0, indicating likely photo or print), 0.2-0.5 for very low motion (2-10, possibly still finger), 0.5-0.7 for low motion (10-30, likely real but still), 0.7-0.9 for moderate motion (30-60, definitely live), and 0.9-1.0 for high motion (above 60, very live). Low motion combined with low variance (below 2.0) receives a penalty of 0.5×, indicating a static spoof. Real fingers typically show 30-100 motion with variance above 2.0, while photos and prints show below 20 motion with variance below 2.0.

Texture-Based Detection

The system employs three methods for texture analysis. Edge pattern analysis computes Sobel gradients and calculates gradient magnitude, then determines the coefficient of variation (standard deviation divided by mean). Very uniform textures (CV below 0.15) score 0.0-0.4 (likely print or photo), natural ranges (0.15-0.8) score 0.4-0.9 (likely real), and very chaotic textures (above 0.8) score 0.75-0.9 (likely real, possibly with noise).

Frequency pattern analysis divides the image into 4×4 patches and analyzes frequency in each patch through ridge crossings. It computes variance of frequencies across patches, as real skin shows varied frequencies while prints and photos show uniform frequencies. Very uniform frequency variance (below 0.03) scores 0.0-0.4, natural ranges (0.03-0.12) score 0.4-0.8, and very varied patterns (above 0.12) score 0.8-1.0.

Texture variance analysis divides the image into 16×16 pixel patches and computes standard deviation for each patch, then averages patch variances. Real skin shows moderate variance (10-60), while prints and photos appear very uniform (below 10) or show artificial patterns. Very uniform textures score 0.0-0.4, natural ranges score 0.4-0.9, and high variance (above 60) scores 0.7-0.9.

The combined texture score weights these three methods:

```
textureScore = (0.5 × edgeScore) + (0.3 × frequencyScore) + (0.2 × varianceScore)
```

Additional heuristics penalize very uniform textures (all scores below 0.2), boost natural variety (any score above 0.35), and provide extra boost when multiple scores are good (at least 2 scores above 0.4).

Consistency Checking

Frame similarity analysis converts frames to grayscale and computes normalized correlation between consecutive frames. High similarity suggests a static object like a photo or print. Extremely high similarity (above 0.99) scores 0.1-0.3 (likely photo), high similarity (0.95-0.99) scores 0.4-0.7 (possibly still finger), moderate-high (0.90-0.95) scores 0.6-0.8 (likely real but still), moderate (0.85-0.90) scores 0.6-0.75 (good variation), and low similarity (below 0.85) scores 0.6-0.75 (lots of variation).

The system adjusts scores based on variance: very high similarity (above 0.99) with very low variance (below 0.001) receives a 0.6× penalty, while high similarity (0.95-0.99) with some variance (above 0.01) receives a 1.1× boost.

Compression Artifact Detection

This method detects JPEG compression artifacts in gallery photos through three approaches. Color quantization detection converts images to HSV color space and analyzes the value (brightness) channel. JPEG compression creates fewer unique colors through quantization. Very few unique colors (below 20%) score 0.15 (heavy JPEG), few unique colors (20-35%) score 0.3, moderate (35-50%) scores 0.45, some quantization (50-65%) scores 0.6, and natural variation (above 65%) scores 0.75. The system also detects banding by counting large color jumps (above 20). High banding ratios (above 15%) apply a 0.8× penalty, while moderate banding (10-15%) applies a 0.9× penalty.

Block uniformity detection divides images into 8×8 blocks (JPEG block size) and computes variance for each block, then calculates variance of variances. JPEG creates uniform variance across blocks. Extremely uniform patterns (below 20) score 0.2 (likely JPEG), very uniform (20-60) scores 0.2-0.3, moderate variation (60-150) scores 0.3-0.65, good variation (150-300) scores 0.65-0.8, and high variation (above 300) scores 0.8 (natural).

Unnatural edge detection uses Canny edge detection and checks for regular edge patterns in 8×8 blocks. JPEG creates edges at block boundaries. Very regular edges (above 80% blocks) score 0.2, regular edges (60-80%) score 0.4, and natural edges (below 60%) score 0.7.

The combined compression score applies different weightings based on indicator strength. Very strong indicators (multiple methods below 0.2) receive a score of 0.15, strong indicators (any method below 0.25) use weighted minimum, and otherwise the system calculates: 0.4 × quantization + 0.3 × blocks + 0.3 × edges.

Screen Pattern Detection

This method detects photos displayed on screens. Regular pattern detection computes gradient magnitude and samples horizontal and vertical lines, using autocorrelation to detect periodicity. Screens create regular pixel grid patterns. High periodicity (above 0.7) indicates a strong screen pattern, while moderate values (0.5-0.7) suggest a possible screen pattern.

Moiré detection analyzes gradients for wave-like patterns that result from screen interference, using frequency domain analysis. Frequency-based moiré detection divides images into 64×64 patches and uses autocorrelation to detect periodicity from screen interference patterns.

The combined screen score evaluates indicator strength. Very strong indicators (multiple methods) score 0.15, strong patterns (horizontal or vertical above 0.7