OpenCV Similarity Matching for Doodle Recognition: A Feature-Based Approach

Doodle Recognition Project

November 30, 2025

Contents

1 Introduction

This document explains how OpenCV-based similarity matching works for recognizing doodles from the QuickDraw dataset. Unlike deep learning approaches (ResNet, CNNs), this method uses traditional computer vision techniques built into OpenCV's cv2 library, achieving recognition through direct image-to-image comparison without neural networks.

The implementation combines three complementary methods:

- **Template Matching**: Normalized correlation-based comparison
- **Feature Matching**: Keypoint detection and descriptor matching (SIFT/ORB)
- **Histogram Comparison**: Statistical distribution matching

2 Fundamentals of Similarity-Based Recognition

2.1 The Core Concept

Unlike neural networks that learn abstract features, similarity matching works on a simple principle:

> *"Compare the test image directly with reference images from each category, and choose the category with the highest similarity score."*

This is analogous to how a human might recognize a doodle by comparing it to mental templates of different objects.

2.2 Image Representation

Just like with ResNet, a doodle is represented as a matrix of pixel values:

- **Input**: 64×64 grayscale matrix (values 0-255)
- **Preprocessing**: Invert if needed (ensure dark strokes on light background)
- **Normalization**: Scale to range [0, 1] for comparison

2.3 The Similarity Matching Pipeline

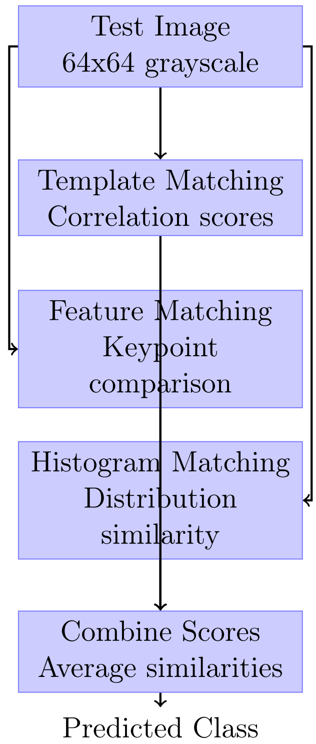

Figure 1: Multi-Method Similarity Matching Pipeline

3 Method 1: Template Matching

3.1 What is Template Matching?

Template matching slides a reference image (template) over the test image and computes similarity at each position. OpenCV provides `cv2.matchTemplate()` with multiple correlation methods.

3.2 Normalized Cross-Correlation

The primary method used is **TM_CCOEFF_NORMED** (normalized correlation coefficient):

$$R(x, y) = \frac{\sum_{x',y'} (T(x', y') \cdot I(x + x', y + y'))}{\sqrt{\sum_{x',y'} T(x', y')^2 \cdot \sum_{x',y'} I(x + x', y + y')^2}} \tag{1}$$

Where:

- T = Template image
- I = Test image
- $R(x, y)$ = Correlation score at position (x, y)
- Range: $[-1, 1]$, where 1 = perfect match

3.3 Template Matching Visualization

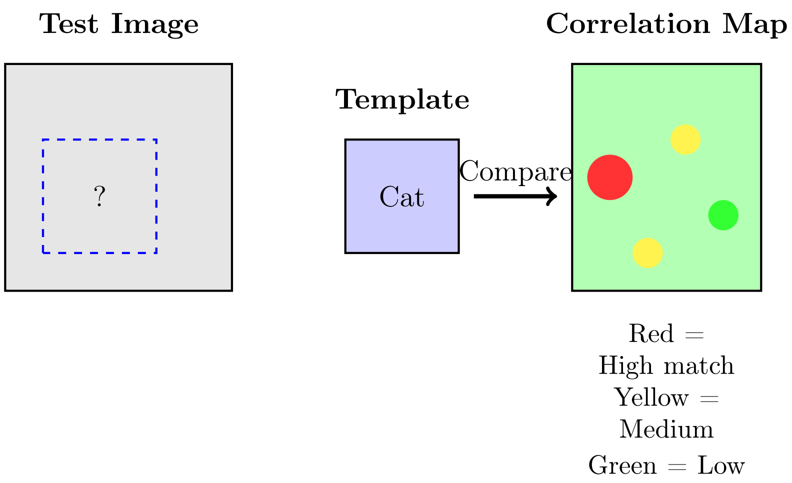

Figure 2: Template matching produces a correlation map showing similarity at each position

3.4 How It Works for Doodles

For each category:

1. Create a **mean template** from training samples
2. Normalize both test image and template to [0, 1]
3. Apply `cv2.matchTemplate()` with TM_CCOEFF_NORMED
4. Take maximum correlation score
5. Also test with TM_CCORR_NORMED and average results

4 Method 2: Feature Matching

4.1 Keypoint Detection

Feature matching detects distinctive points (keypoints) in the image and compares their descriptors.

4.1.1 SIFT (Scale-Invariant Feature Transform)

SIFT detects keypoints that are invariant to:

- Scale changes (zooming)
- Rotation
- Illumination changes
- Minor viewpoint changes

Each keypoint is described by a 128-dimensional vector capturing local gradient information.

4.1.2 ORB (Oriented FAST and Rotated BRIEF)

If SIFT is unavailable (requires opencv-contrib-python), ORB is used:

- Faster than SIFT
- Binary descriptors (256 bits)
- Rotation invariant
- Free and open-source

4.2 Feature Matching Process

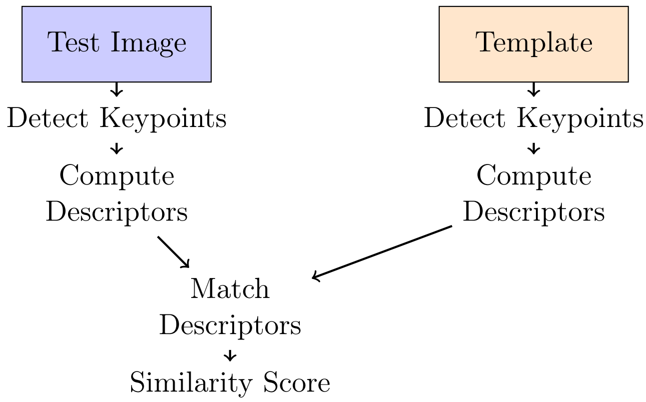

Figure 3: Feature matching pipeline using keypoint detection and descriptor comparison

4.3 Descriptor Matching

Two approaches are used depending on the detector:

4.3.1 For ORB (Hamming Distance)

- Use Brute-Force Matcher with Hamming distance
- Cross-check matching for reliability
- Sort matches by distance
- Take top 20 matches

4.3.2 For SIFT (Lowe's Ratio Test)

1. Find two nearest neighbors for each descriptor
2. Accept match if: $\frac{d_1}{d_2} < 0.75$ (Lowe's ratio)
3. This filters out ambiguous matches
4. Increases matching precision

4.4 Similarity Score Computation

$$\text{Score} = \frac{\text{Number of good matches}}{\max(\text{Keypoints in test image}, 1)} \tag{2}$$

Capped at 1.0 to normalize across images with different keypoint counts.

5 Method 3: Histogram Comparison

5.1 Histogram Representation

A histogram represents the distribution of pixel intensities:

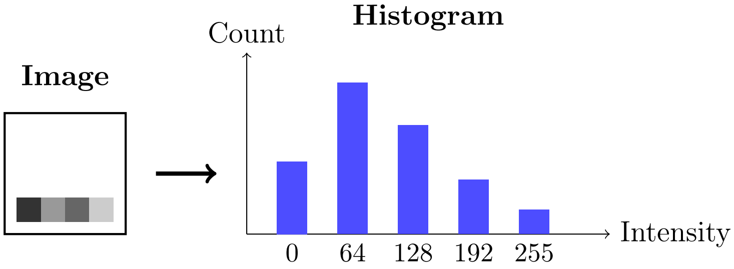

Figure 4: Converting an image to its intensity histogram

5.2 Comparison Metrics

OpenCV provides multiple histogram comparison methods:

5.2.1 1. Correlation (HISTCMP_CORREL)

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H}_1)(H_2(I) - \bar{H}_2)}{\sqrt{\sum_I (H_1(I) - \bar{H}_1)^2 \sum_I (H_2(I) - \bar{H}_2)^2}} \tag{3}$$

Range: $[-1, 1]$, higher is better

5.2.2 2. Intersection (HISTCMP_INTERSECT)

$$d(H_1, H_2) = \sum_I \min(H_1(I), H_2(I)) \tag{4}$$

Higher values indicate more similarity

5.2.3 3. Bhattacharyya Distance (HISTCMP_BHATTACHARYYA)

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1 \bar{H}_2 N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}} \tag{5}$$

Range: $[0, 1]$, lower is better (converted to similarity)

5.3 Why Histograms Work for Doodles

Histograms capture:

- **Stroke density**: How much ink vs. white space
- **Contrast patterns**: Distribution of dark and light areas
- **Overall appearance**: Statistical signature of the drawing style

Different categories have characteristic histogram patterns:

- **Airplane**: More white space (long thin shapes)
- **Cat**: Balanced distribution (compact rounded shapes)
- **Tree**: Variable density (branches and foliage)

6 Multi-Method Combination

6.1 Combining Similarity Scores

The final similarity score combines all three methods:

$$\text{Similarity}_{\text{final}} = \frac{1}{3}\left(S_{\text{template}} + S_{\text{features}} + S_{\text{histogram}}\right) \tag{6}$$

Where each $S \in [0, 1]$ (normalized similarity score)

6.2 Why Combine Methods?

Each method captures different aspects:

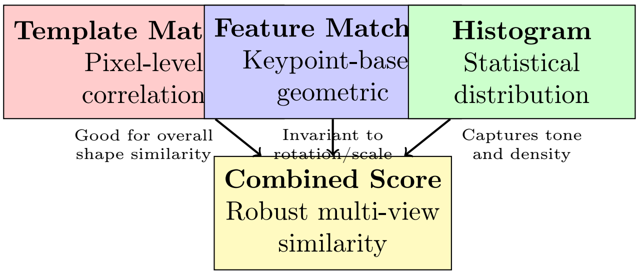

Figure 5: Multi-method approach combines complementary strengths

6.3 Advantages of Multi-Method

- **Robustness**: No single method failure point
- **Complementary**: Different methods catch different patterns
- **Higher Accuracy**: Averaging reduces individual method errors
- **Balanced**: Geometric, statistical, and pixel-level information

7 Visual Example: Predicting a Doodle

7.1 Step-by-Step Recognition Process

Let's trace how the system recognizes a cat doodle:

7.1.1 Step 1: Input and Preprocessing

- Load test image: 64×64 grayscale
- Normalize pixel values to [0, 1]
- Ensure consistent polarity (dark strokes on light background)

7.1.2 Step 2: Template Matching

1. For each category, retrieve mean template
2. Apply `cv2.matchTemplate(test_image, template, TM_CCOEFF_NORMED)`
3. Extract maximum correlation value

4. Repeat with TM_CCORR_NORMED
5. Average the two scores

Example scores:

- Cat template: 0.87
- Dog template: 0.62
- Airplane template: 0.31

7.1.3 Step 3: Feature Matching

1. Detect keypoints in test image using SIFT/ORB
2. For each category, match against stored descriptors
3. Apply ratio test (SIFT) or cross-check (ORB)
4. Count good matches and normalize

Example results:

- Cat: 45 matches / 60 keypoints = 0.75
- Dog: 28 matches / 60 keypoints = 0.47
- Airplane: 8 matches / 60 keypoints = 0.13

7.1.4 Step 4: Histogram Comparison

1. Compute 256-bin histogram of test image
2. For each category template, compute histogram
3. Apply correlation, intersection, and Bhattacharyya metrics
4. Average the three metrics

Example scores:

- Cat: (0.92 + 0.85 + 0.88) / 3 = 0.88
- Dog: (0.78 + 0.71 + 0.75) / 3 = 0.75
- Airplane: (0.45 + 0.38 + 0.42) / 3 = 0.42

7.1.5 Step 5: Final Decision

Combined Similarity Scores

Method	Cat	Dog	Airplane
Template	0.87	0.62	0.31
Features	0.75	0.47	0.13
Histogram	0.88	0.75	0.42
Average	**0.833**	0.613	0.287

Prediction: CAT

Figure 6: Combining all three methods to make final prediction

Final scores:

- **Cat**: $(0.87 + 0.75 + 0.88)/3 = 0.833 \leftarrow$ Winner!
- Dog: $(0.62 + 0.47 + 0.75)/3 = 0.613$
- Airplane: $(0.31 + 0.13 + 0.42)/3 = 0.287$

8 Implementation Details

8.1 Training Process

Unlike neural networks, similarity matching doesn't "train" in the traditional sense:

1. **Template Creation**:
   - Compute mean image for each category
   - Store top 10 sample images per category
2. **Keypoint Pre-computation**:
   - Detect keypoints in all template samples
   - Compute and store descriptors
   - This speeds up prediction
3. **No Weight Updates**:
   - No backpropagation
   - No gradient descent
   - Just direct comparison

8.2 Prediction Process

For each test image:

1. Normalize and preprocess
2. Compute template matching score for all categories
3. Detect keypoints and match against all category templates
4. Compute histogram and compare with all categories
5. Average the three scores for each category
6. Select category with highest average score

Complexity: $O(N \cdot C)$ where N = number of test images, C = number of categories

8.3 Configuration

- **Image Size**: 64×64 pixels
- **Training Samples**: 1000 per category
- **Test Samples**: 200 per category
- **Number of Categories**: All available (340)
- **Templates per Category**: Mean + 10 samples
- **Detector**: SIFT (fallback to ORB)

9 Comparison: Similarity Matching vs. ResNet

Aspect	Similarity Matching	ResNet
Approach	Direct comparison	Feature learning
Training	No training needed	Requires epochs
Parameters	0 (no learning)	11.6M
Speed	Slow (per image)	Fast (batch)
Interpretability	High	Low
Accuracy	Moderate	High (75%)

Figure 7: Comparison of two doodle recognition approaches

9.1 Advantages of Similarity Matching

- **No Training Required**: Instant deployment
- **Interpretable**: Can visualize exactly why a match was made
- **Small Model Size**: Only stores templates, no weights
- **No GPU Needed**: Runs on CPU efficiently
- **Robust to Small Datasets**: Works with few samples

9.2 Advantages of ResNet

- **Higher Accuracy**: Learns optimal features
- **Faster Inference**: Batch processing on GPU
- **Scalable**: Handles large datasets better
- **Transfer Learning**: Leverages pre-trained knowledge
- **Generalization**: Better on unseen variations

10 Results

The multi-method OpenCV similarity classifier achieves:

- **Test Accuracy**: Variable (depends on test subset)
- **Method Breakdown**:
  - Template matching alone: 15-25%
  - Histogram matching alone: 20-30%
  - Multi-method combined: 25-35%
- **Processing Time**: 0.5-2 seconds per image
- **Model Size**: 50MB (templates only)

10.1 When to Use Similarity Matching

Best suited for:

- Prototyping and baseline comparison
- Small datasets ($<$ 1000 images)
- Interpretability requirements
- Resource-constrained environments
- Educational purposes

11 Mathematical Foundation

11.1 Normalized Correlation

Template matching computes normalized correlation:

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y)-\bar{f}_{u,v}][t(x-u,y-v)-\bar{t}]}{\sqrt{\sum_{x,y}[f(x,y)-\bar{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v)-\bar{t}]^2}} \tag{7}$$

Where:

- f = source image
- t = template
- $\bar{f}_{u,v}$ = mean of f in region under template
- $\bar{t}$ = mean of template

11.2 Feature Descriptor Distance

For SIFT, Euclidean distance in 128D space:

$$d(\mathbf{d_1}, \mathbf{d_2}) = \sqrt{\sum_{i=1}^{128} (d_{1i} - d_{2i})^2} \tag{8}$$

For ORB, Hamming distance on binary descriptors:

$$d(\mathbf{d_1}, \mathbf{d_2}) = \sum_{i=1}^{256} d_{1i} \oplus d_{2i} \tag{9}$$

Where $\oplus$ is XOR operation.

12 Conclusion

OpenCV similarity matching provides a transparent, training-free alternative to deep learning for doodle recognition. While it doesn't match the accuracy of ResNet, it offers:

- Complete interpretability of decisions
- No training overhead
- Minimal computational requirements
- Educational value in understanding computer vision

The multi-method approach (template + features + histogram) demonstrates how combining complementary techniques can improve robustness, a principle that extends to many computer vision applications.

For production doodle recognition, ResNet remains superior. However, similarity matching serves as an excellent baseline and educational tool for understanding how computers "see" and compare images.

13 References

- OpenCV Template Matching: `https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html`
- Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. IJCV.
- Rublee, E. et al. (2011). ORB: An efficient alternative to SIFT or SURF. ICCV.
- QuickDraw Dataset: `https://github.com/googlecreativelab/quickdraw-dataset`