SIMC2024Report

Li Shiming xajdfz
Liu Ye xajdfz
Wang Boran xajdfz

Abstract

In this paper, we propose an efficient algorithmic approach to classify image patterns based on their geometric orientation. We address two scenarios: standard 2D image matrices (Task 1) and flattened 1D feature vectors (Task 2). Our method utilizes a "Generate and Match" strategy, where a reference image is rotated to create a template dictionary. For high-dimensional flattened data, we introduce a spatial reconstruction step to enable accurate rotation. The experimental results verify the accuracy and robustness of the proposed method.

1 Introduction

Pattern recognition tasks often require identifying the orientation of an object relative to a reference. The dataset provided consists of patterns rotated at discrete angles: $0°, 90°, 180°$, and $270°$.

The challenge lies in:

1. **Task 1:** Handling 2D matrix data directly.
2. **Task 2:** Handling 1D flattened vectors where spatial information is not explicitly available.

2 Methodology

2.1 Template Generation Strategy

Since the possible rotations are limited to a finite set $S = \{0°, 90°, 180°, 270°\}$, we avoid complex feature extraction. Instead, we employ a reference-based approach: 1. Select the first image as the *Reference Pattern*. 2. Apply the rotation function $Rot(img, k)$ where $k \in \{0, 1, 2, 3\}$. 3. Store the 4 variations in a dictionary: $D = \{angle : pattern\}$.

2.2 Handling Flattened Data (Task 2)

In Task 2, the data is given as vectors of length $L = 1296$. A vector cannot be rotated geometrically. Therefore, we implement a **Reshape-Rotate-Flatten** pipeline:

$$Dim = \sqrt{L} = \sqrt{1296} = 36 \tag{1}$$

The process is defined as:

$$Vector_{1D} \xrightarrow{\text{Reshape}} Matrix_{2D} \xrightarrow{\text{Rotate}} Matrix'_{2D} \xrightarrow{\text{Flatten}} Vector'_{1D}$$

This ensures that the rotation respects the spatial topology of the original image.

3 Algorithm Description

The complete logic is described in Algorithm 1. This algorithm is universal for both tasks, with a conditional branch for data reshaping.

Algorithm 1 Orientation Classification Algorithm

Input: Dataset patterns P, Reference Image R
Output: Count Dictionary $Counts$
Initialize $Counts \leftarrow \{0:0, 90:0, 180:0, 270:0\}$
Initialize $Templates \leftarrow \{\}$
▷ Step 1: Build Template Library
for $k = 0$ to 3 **do**
    $Angle \leftarrow k \times 90$
    $RotatedImg \leftarrow Rotate(R, k)$
    if Task is Task 2 **then**
        $Templates[Angle] \leftarrow Flatten(RotatedImg)$
    else
        $Templates[Angle] \leftarrow RotatedImg$
    end if
end for
▷ Step 2: Matching Process
for each pattern p in P **do**
    for $angle, template$ in $Templates$ **do**
        if $p \approx template$ **then** ▷ Check similarity
            $Counts[angle] \leftarrow Counts[angle] + 1$
            break
        end if
    end for
end for
return $Counts$

4 Implementation and Results

We implemented the algorithm using Python with `NumPy` for efficient matrix operations.

4.1 Task 1 Results

For the 2D matrix data, the algorithm successfully matched every pattern to the reference set. The computational cost was minimal due to the small image size (10×10).

4.2 Task 2 Results

For the flattened data ($L = 1296$), the reshaping strategy worked perfectly. By converting the vectors back to 36×36 matrices, we preserved the geometric integrity during rotation.

5 Conclusion

We presented a robust solution for discrete orientation classification. The key contribution is the handling of flattened vector data by temporarily restoring its spatial dimensions. This method is exact, efficient, and easily extensible to other geometric transformations.

6 AI Use Report

Trae IDE was uesed for inline code completions.**Gemini 3.0** was used to generate LaTeX code in formatting the document