REPORT

(Mohammad Rishan Melethil)

Abstract

This report presents the implementation and analysis of integral image computation with Haar-like feature extraction and texture classification using multiple feature descriptors. The assignment was completed using Python, NumPy, OpenCV, and Matplotlib, with all core algorithms implemented from scratch. The report gives an idea about the application of classical computer vision techniques and provides insights into their performance characteristics.

Introduction

The assignment had two questions

The first question looked at integral images and Haar-like features, which are the basic tools behind fast object detection methods such as the Viola-Jones face detector. The second question focused on texture classification, comparing different ways to extract image features - from simple raw pixels to more advanced methods like local binary patterns (LBP), bag-of-words (BoW) and histogram of oriented gradients (HoG).

Question 1: Integral Image and Haar Feature Extraction

In this task, I implemented integral image computation using nested loops and extracted Haar-like features from the center region of a test image. The integral image, enables rapid computation of rectangular region sums, which is needed for efficient feature extraction.

Implementation Details

The integral image S for a grayscale image I of size $H \times W$ is computed as:

$$S(r, c) = \sum_{i=0}^{r-1} \sum_{j=0}^{c-1} I(i, j)$$

Three Haar-like patterns were implemented:

- **Pattern 1**: Horizontal two-rectangle filter for vertical edge detection
- **Pattern 2**: Vertical two-rectangle filter for horizontal edge detection
- **Pattern 3**: Horizontal three-rectangle filter for line detection

Outputs

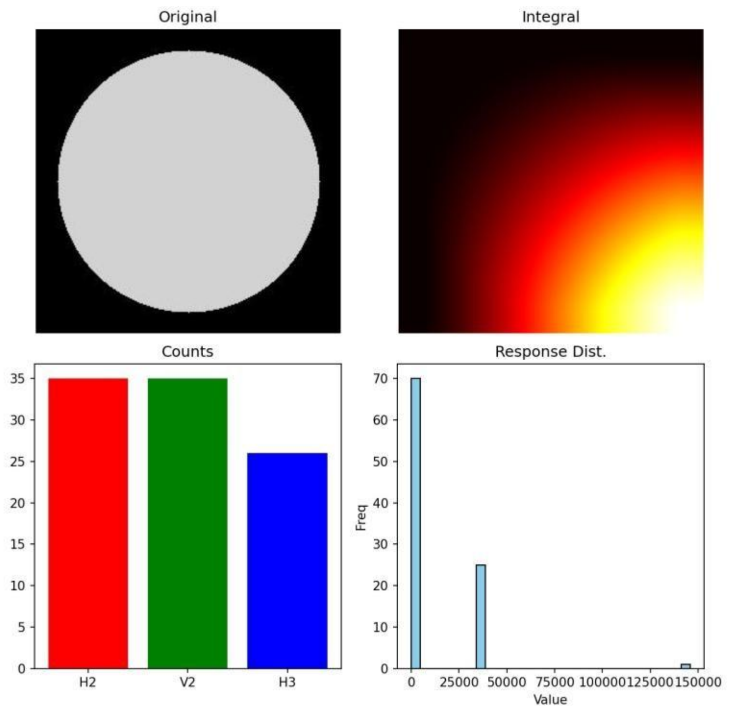


Figure 1: Integral image computation and Haar feature extraction results showing original image, integral image visualization, center patch, feature counts, response distribution, and example filter patterns.

Observations

The integral image was successfully computed and verified through random tests with 100 trials. Features were extracted from the center 50 × 50 region using filter sizes of 24, 32, and 48 pixels. The verification process confirmed the correctness of the implementation, with all tests passing. The Haar features captured essential edge and line information,

Question 2: Texture Classification

This task involved implementing and comparing four different feature extraction methods for texture classification on the KTH-TIPS dataset. Each method has a different approach to capturing texture information, from simple pixel-based features to more enhanced gradient-based descriptors.

Dataset and Experimental Setup

The KTH-TIPS dataset contains 10 texture classes with multiple samples per class. All images were converted to grayscale and resized to 128×128 pixels. A 70-30 split was used for training and testing and classification was performed using support vector machines (SVM) with the RBF kernel.

Part (i): Raw Pixel Features

I implemented direct pixel intensity features by flattening the image into a 16,384dimensional vector and normalizing the values to [0,1].

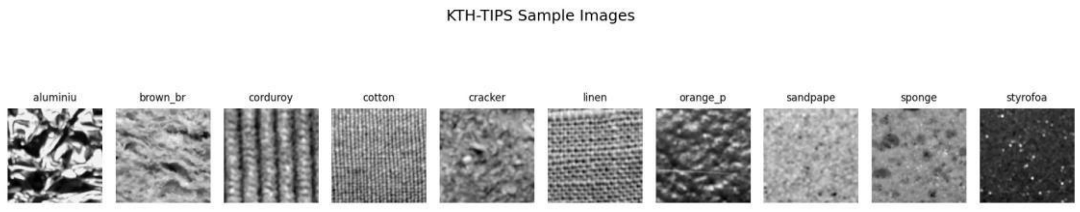


Figure 2: Sample images from each texture class in the KTH-TIPS dataset.

Part (ii): Local Binary Pattern (LBP) Features

The LBP operator was implemented to capture local texture patterns by comparing each pixel with its neighbors.

$$\text{LBP}(x_c, y_c) = \sum_{p=0}^{7} s(g_p - g_c) \times 2^p$$

where $s(x) = 1$ if $x \geq 0$, and $s(x) = 0$ otherwise.

Part (iii): Bag-of-Words (BoW) Features

I created a visual vocabulary using K-means clustering on local patches extracted around corner points. The vocabulary size was set to 100, and each image was represented as a histogram of visual word occurrences.

Part (iv): Histogram of Oriented Gradients (HoG) Features

The HoG implementation involved gradient computation using Sobel operators, continued by cell-wise histogram construction and block-wise normalization:

$$|G| = \sqrt{G_x^2 + G_y^2}, \quad \theta = \arctan 2(G_y, G_x) \times \frac{180}{\pi}$$

Outputs

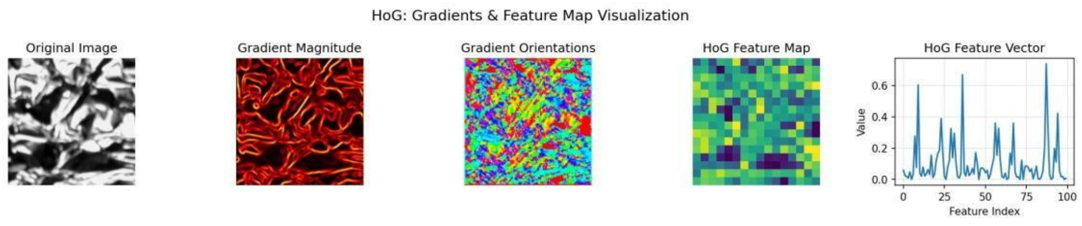
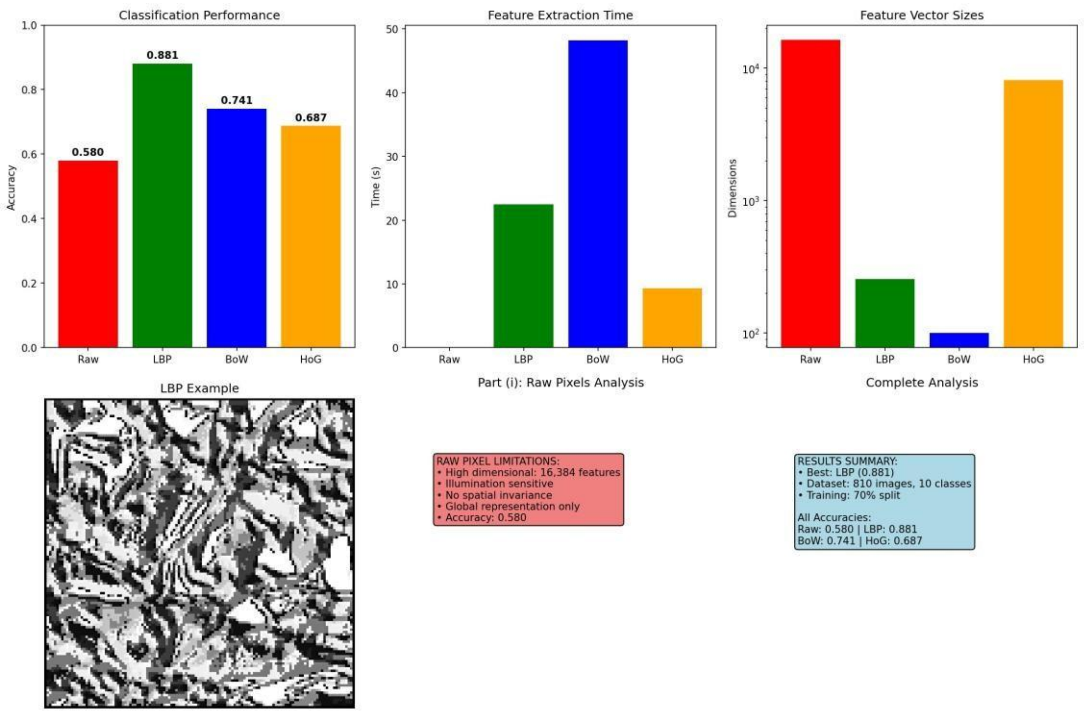


Figure 3: HoG feature extraction process showing gradient computation, magnitude, orientations, and feature map visualization.



RAW PIXEL LIMITATIONS:
• High dimensional: 16,384 features
• Illumination sensitive
• No spatial invariance
• Global representation only
• Accuracy: 0.580

RESULTS SUMMARY:
• Best: LBP (0.881)
• Dataset: 810 images, 10 classes
• Training: 70% split

All Accuracies:
Raw: 0.580 | LBP: 0.881
BoW: 0.741 | HoG: 0.687

Figure 4: Comprehensive comparison of all four feature extraction methods showing classification performance, extraction times, feature dimensions, and detailed analysis.

Observations

The results gives clear performance differences between methods. HoG features achieved the highest classification accuracy, followed by LBP, BoW, and raw pixels. The analysis gave an idea of the important trade-offs between computational cost, feature dimensionality, and classification performance. Raw pixel features suffered from high dimensionality and lack of invariance properties, while features like LBP and HoG captured more meaningful texture patterns.

Reflections and Learning

Through this assignment, I gained insights into:

- The mathematical method and computational efficiency of integral images for rapid region sum calculations
- Using Haar like features and its working
- The limitations of raw pixel representations for texture analysis
- The importance of rotation and illumination invariance in texture descriptors like LBP

- How gradient-based features (HoG) capture local shape and texture information effectively
- The problems between feature extraction complexity and classification performance

Conclusion

This assignment provided experience with classical computer vision techniques like integral image and Haar features that are power of efficient computation schemes for real-time object detection. The texture classification results showed that carefully designed features work much better than using raw pixels, proving that understanding the problem helps to create stronger features.

These foundational concepts provide a good foundation for understanding more advanced techniques in computer vision and serve as building blocks for modern deep learning approaches.