Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Computer Vision

ENCS5343

Assignment No.2

Arabic Handwritten Text Identification Using Local Feature Extraction Techniques

Prepared by:

Qusay Taradeh **1212508.**

Karim Marayta **1211610.**

Instructor: **Dr. Aziz Qaroush.**

Section: **1.**

Date: **Monday, December 23, 2024.**

Table of Contents

Table of Contents i

Table of Figures ii

Introduction 1

Experiments And Results 1

Step 1: Loading Dataset 1

Step 2: Transformation Images 1

Step 3: Preprocessing Images 2

Step 4: Feature Extraction with Keypoint Descriptors 2

Step 5: Clustering Descriptors into Visual Vocabulary 2

Step 6: Constructing Feature Vectors 3

Step 7: Model Training 3

Step 8: Testing and Evaluation 3

Step 9: Visualization 3

Step 10: Measuring Performance 3

Average number of features extracted, time taken, and accuracy 4

Mean Accuracy by transformation type and heatmap accuracy 4

Features extracted by descriptor 5

Discussion: 5

Normal test set results 5

Average number of features extracted, time taken, and accuracy 5

Transformed test set results 5

Mean Accuracy by transformation type and heatmap accuracy 5

Features extracted by descriptor 6

Conclusion: 6

Table of Figures

Figure 1: Transformed Images....2
Figure 2: Accuracy by descriptors for various number of clusters 3
Figure 3: Average number of features. Figure 4: Time taken by descriptor on the train dataset. 4
Figure 5: Accuracy by descriptor 4
Figure 6: Mean Accuracy for each transformation type. Figure 7: Accuracy heatmap between descriptors and transformed test set 4
Figure 8: Features extracted....5

Introduction

Handwritten text identification is a challenging problem in computer vision, particularly for scripts like Arabic, which feature complex shapes and styles. Local feature extraction techniques, such as Scale-Invariant Feature Transform (SIFT), are crucial in detecting and describing key points in images that remain consistent despite variations in scale, rotation, and illumination. This assignment focuses on exploring these techniques by comparing the SIFT algorithm with other related methods like Oriented FAST and Rotated BRIEF (ORB) and AKAZE.

The process involves preprocessing handwritten text samples, extracting local features, and matching key points between images. Performance will be evaluated based on key metrics such as accuracy, efficiency, and robustness to variations in scale, rotation, illumination, blur, and noise. By analyzing these aspects, the assignment emphasizes the significance of choosing the right features and balancing computational cost with accuracy.

Using the AHAWP dataset, which provides a diverse set of Arabic handwritten word samples, this assignment aims to deepen the understanding of local feature extraction and its applications in real-world problems. Through implementation and evaluation, we aim to identify the strengths and limitations of each algorithm, gaining insights into their suitability for handwritten text identification.

Experiments And Results

The experiments done with 4 steps as following:

Step 1: Loading Dataset

The dataset containing user images were loaded as greyscale into the program. Each user's images were read from a directory using OpenCV's imread function. Afterward, the dataset was divided into two parts: 80% for training and 20% for testing. This division allows us to train the model on the majority of the data while reserving a portion to evaluate its performance.

Step 2: Transformation Images

In this step the transformation or variation added to the test set with multiple values for each type as following:

Scale: **80%** as zoom out, and **120%** as zoom in.

Rotation: -30, -60, -90, 30, 60, 90 degrees.

Illumination: 0.1, 0.5 darken to **1.5, 1.9** brighten.

Gaussian Noise and Blur: σ = 25, 50, 100.

And the results on 5 different images as following in figures below with variation type on left corner of each row:

Figure 1: Transformed Images

Then, in next steps the process applied on **both original** test set and **variated test set.**

Step 3: Preprocessing Images

Before training the model, image preprocessing was attempted to enhance their features. Techniques like thresholding using Otsu's method, and Canny edge detection (upper threshold = Otsu's threshold, lower threshold = 0.5*upper threshold) have been attempted, but resulted in losing important identification information, as concluded from the substantially worsened performance compared to unprocessed images. Thus, we decided to leave the images as is.

Step 4: Feature Extraction with Keypoint Descriptors

Keypoints and their descriptors were extracted from each image using a specified feature detector (SIFT or ORB or AKAZE). These descriptors capture local image characteristics, making them suitable for tasks like object recognition or clustering. The descriptors for all images were combined into a single dataset for clustering.

As clustering greatly reduced the dimensionality of the extracted features, we decided to not limit the detectors in the number of keypoints per image. It would also interfere with the comparison of the detectors in terms of the number of detected keypoints.

Step 5: Clustering Descriptors into Visual Vocabulary

The extracted descriptors were grouped using the mini batch kmeans clustering algorithm. The clusters formed a visual vocabulary (Bag of Words) where each cluster represents a unique feature. The best number of clusters has been obtained using 5-fold cross validation with tested values [500, 1000, 2000, 4000] and the best number approximately was **2000** for **SIFT**, and **1000** for all other descriptors as following in figure below:

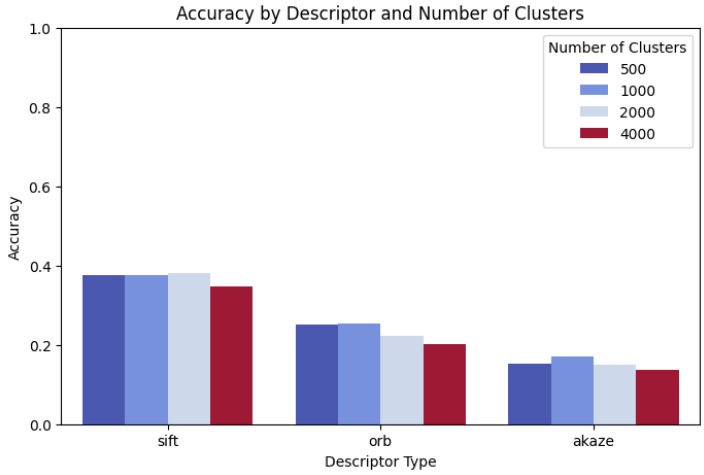

Figure 2: Accuracy by descriptors for various number of clusters

Step 6: Constructing Feature Vectors

For each image, a histogram of cluster occurrences (feature vector) was computed. This histogram describes the distribution of visual words (clusters) in the image, effectively representing it in a structured numerical format suitable for machine learning models.

Step 7: Model Training

The feature vectors were scaled using StandardScaler to normalize their values using Z-score. A classifier (Random Forest or SVM) was then trained on the normalized feature vectors and their corresponding labels (user IDs) with multiple number of estimators for Random Forest between these values [500, 750, 1000] and C variable for SVM with RBF kernel between these values [1, 10, 20]. Using GridSearch to find the best hyperparameters, the **number of estimators = 1000** for **Random Forest** and **C = 10** for **SVM** for all descriptors. Then, creating any of (SIFT or ORB or AKAZE) feature detectors and training it with the cluster and its parameters. This training process enabled the model to learn patterns from the dataset.

Step 8: Testing and Evaluation

The trained model was tested on the 20% reserved dataset. Predictions were made on the test images, and the accuracy of the model was evaluated by comparing predicted labels to actual user IDs. Also, the testing done on transformed images set that could be scaled, noisy, rotated, and illuminated. This step provided insights into the model's performance and its ability to generalize to unseen data with natural variations.

Step 9: Visualization

In this step the results of each detector such as accuracy, average number of features, calculating descriptors time are stored and visualized in bar plots and heatmap to compare the performance of each one.

Step 10: Measuring Performance

After training all models and evaluating each model on original test images set with best parameters, the average number of features for each image, time of calculating descriptors, and accuracy shown as following in figures below:

Average number of features extracted, time taken, and accuracy

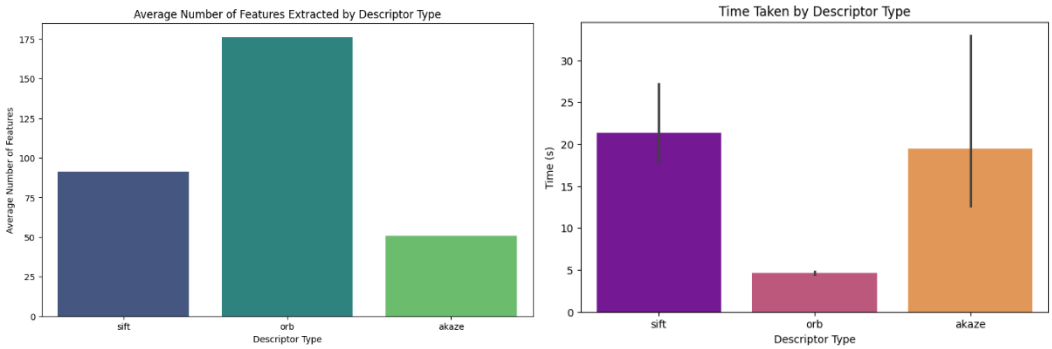

Figure 3: Average number of features. Figure 4: Time taken by descriptor on the train dataset.

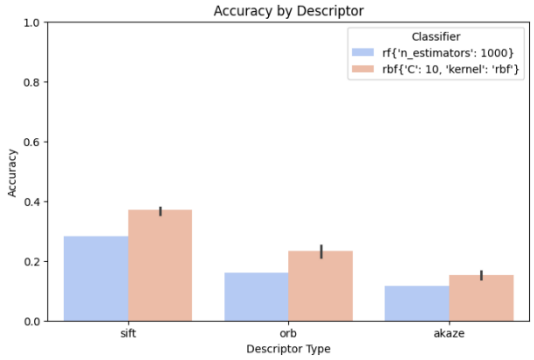

Figure 5: Accuracy by descriptor

Then, after evaluating on **transformed images test** set the results as shown in figures below:

Mean Accuracy by transformation type and heatmap accuracy

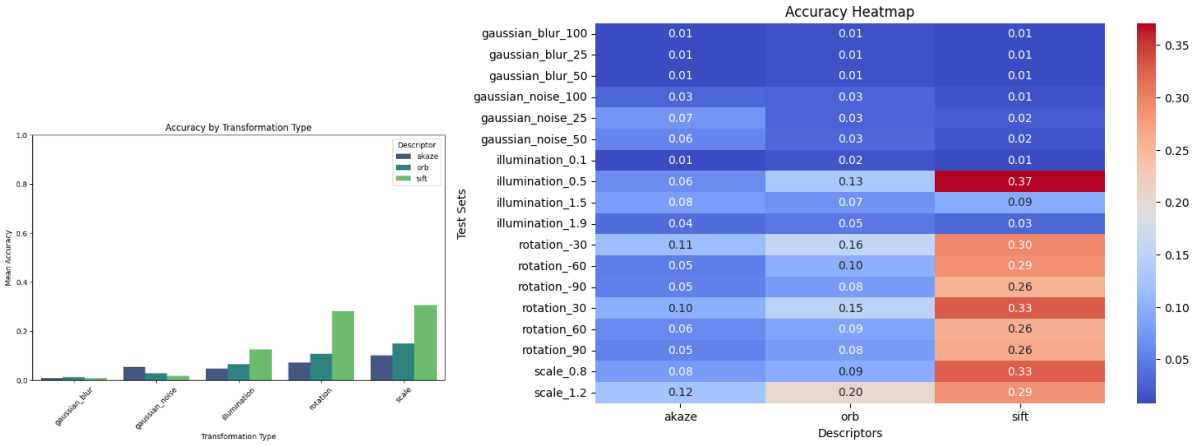

Figure 6: Mean Accuracy for each transformation type. Figure 7: Accuracy heatmap between descriptors and transformed test set

Features extracted by descriptor

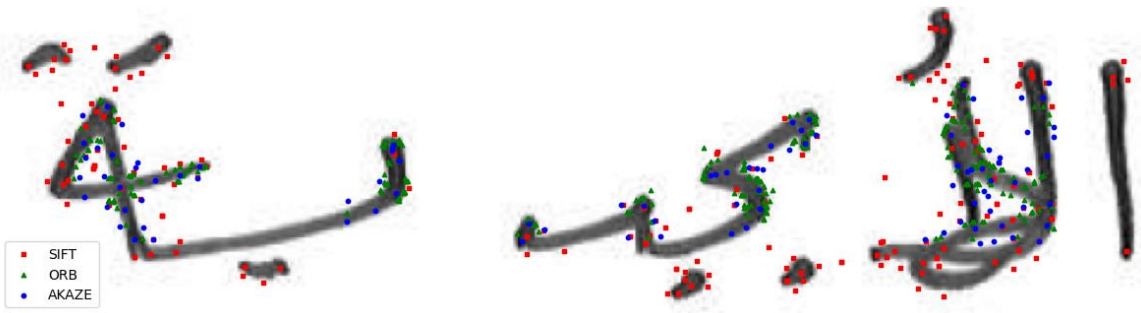

Figure 8: Features extracted

Discussion:

Normal test set results

Average number of features extracted, time taken, and accuracy

Average Number of Features Extracted by Descriptor and Classifier: We notice that the **ORB descriptor** consistently extracts the highest number of features on average across all classifier configurations. However, the features that ORB produces are simpler (binary). On the other hand, the **AKAZE descriptor** extracts the least number of features, suggesting a more selective feature identification process. **SIFT** produces a moderate number of features compared to **ORB** and **AKAZE**. Although the quantity of the features is important, the quality and relevance of these features is even more important.

Time Taken by Descriptor and Classifier: We observe that **AKAZE takes significantly more time** compared to both SIFT and ORB. This is a crucial finding as the computational efficiency of AKAZE is lower than the other two. **ORB is the fastest**, indicating that its algorithm is well-optimized for quick feature detection and description. **SIFT falls in between**, offering a balance between computational speed and accuracy.

Accuracy by Descriptor and Classifier: Accuracy results reveal that **SIFT achieves the highest accuracy** with **SVM** (kernel=rbf, C=10). This suggests that SIFT's feature detection quality aligns well with the chosen classifiers. In contrast, **ORB is faster, but shows a noticeable drop in accuracy**, which could imply that while it detects more features, these might not always be as discriminative or relevant for classification. **AKAZE exhibits relatively lower accuracy**, highlighting that its selective feature detection might not always capture the critical features necessary for identification.

Transformed test set results

Mean Accuracy by transformation type and heatmap accuracy

We notice that the accuracy of different transformation types and descriptors shows significant variation. For instance, Gaussian blur seems to have the lowest accuracy, indicating that this type of transformation introduces distortions that severely impact the performance of our descriptors. The transformations that result in only a partial loss of accuracy are scale, rotation, illumination, noise.

SIFT demonstrates the best robustness to scale, rotation and illumination transformations. ORB is trading off robustness for computational efficiency, but still manages to exhibit a degree of robustness to the same transformation types as SIFT. AKAZE demonstrates bad accuracy overall, but seems to be somewhat robust to gaussian noise.

In general, SIFT has the best robustness to scale, rotation, and illumination transformations. ORB should be used if SIFT is too computationally expensive. Regarding AKAZE, the initial accuracy is too low and the computation is quite expensive, so it is not recommended for this task.

Features extracted by descriptor

From the image depicting features extracted, we observe how different algorithms—SIFT, ORB, and AKAZE—perform on the same piece of Arabic text. Each algorithm highlights significant feature points on the text, marked by red squares for SIFT, green triangles for ORB, and blue circles for AKAZE.

ORB detects the highest amount of keypoints, followed by SIFT and ORB, respectively.

Conclusion:

We explored the effectiveness of different feature extraction algorithms—SIFT, ORB, and AKAZE—in identifying handwritten Arabic text. Our results show that ORB consistently extracts the most features and is the fastest, but not always the most accurate. SIFT strikes a balance between speed and accuracy, achieving the highest accuracy with certain classifiers. AKAZE, although slower, is more selective in feature extraction, but the features it extracts are not very suitable for our task as could be concluded from the accuracy.

Comparing their performance on both original and transformed images highlighted that SIFT has the highest robustness to various types of transformations (in particular scale, rotation and illumination). While ORB is less robust to such changes.

To sum up, this assignment has provided us with a deeper understanding of these algorithms and their effectiveness in identifying handwritten Arabic text. It demonstrates how different algorithms handle variations in handwritten text, helping to make informed decisions about which techniques are best suited for specific tasks. Additionally, it emphasizes the significance of balancing computational cost with accuracy, providing valuable insights for real-world applications in computer vision.