Faculty of Engineering & Technology

Electrical & Computer Engineering Department

COMPUTER VISION - ENCS5343

Assignment #2: Detecting Users' Handwritings using SIFT and ORB as feature extractors technique

---

Prepared by:

Jana Herzallah 1201139

Ahmed Zubaidia 1200105

Instructor: Dr. Aziz Qaroush

Section: 1

Date: 09/11/2024

Contents

Table of Figures:

Table of Tables:

Introduction and Objective

Distinguishing handwritings of users can be very necessary in some applications such as bank documents signatures, we can tell if it has been copied or it original. Applying such applications can be done through multiple ways, one way is finding the basic features of the images that has the handwritings and extracting its features using the classical feature extraction ways such as SIFT and ORB that works by finding the number of key points and representing them via descriptors which can be used later to train classifiers to tell features apart.

This assignment aims to use SIFT and another technique such as ORB as feature extractor techniques and build a system that can distinguish between each users handwriting using multiple concepts such as Bag Of Words, and compare between the two algorithms according to their robustness towards data variations as will be shown in the next sections.

Background

1. Bag of words-BOW

It works as a fundamental representation technique for information retrieval, in which the text or images gets broken into pieces that will be considered as unique components of the system and each image will be broken into main features that is considered as codeword and then the frequency of each will be counted and saved into histograms to be used in matching images systems.

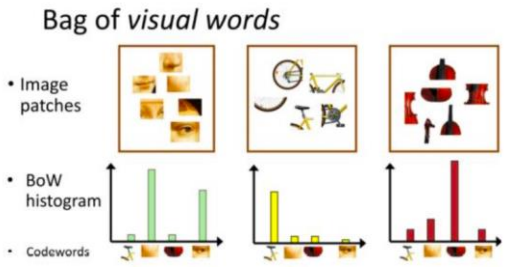

Figure 1: BOW representation on images

2. Grid search parameter tuning

It's an exhaustive technique that works by brute forcing bunch of hyperparameter, it's usually done via training the model on all combinations of the possible parameters and then testing it on validation data and selecting the most robust model according to different metrics such as the accuracy.

3. SIFT

Feature extraction method used in computer vision for identifying features in images. It works by extracting interest points or key points that may be a curve or an edge. This algorithm is invariant to scale, rotation and affine transformations. Each extracted key point is represented by 128-dimensional vector called descriptor. It has outperformed many algorithms but it definitely takes time and resources.

4. ORB

Faster feature extractor model used in real time applications alternative to sift and surf, it only handles rotation and scale invariance and it could result in less accuracy depending on the dataset.

5. SVM

Support vector machines model is a classifier model that works by finding the hyperplane that best separates data points. Its effective in handling linear and non-linear separable data. It handles outliers well and its robust to overfitting. It requires tuning for hyper parameters such as C and kernel parameters.

Methodology and System Structure

1. Environment: The project was hosted under **Google Colab environment** with extended resources up to 51 GB rams instead of the normal rams which are 12.7 GB with T4-GPU processor which is good for intensive tasks. **Concurrent.futures.ProcessPoolExecutor** python library is used to ensure running tasks in parallel and all workload is balanced on the processors.

2. Preprocessing dataset: This phase consisted of multiple steps: **Downloading the AHAWP dataset** form the zipped folder uploaded to the drive and exploring it. It contains 82 users each with 10 trials for each one of the 10 words which are {mehras, abjadiyah,azan, mustadhafeen , qashta, sakhar, shateerah , ghaleez , ghazal, fasyakfeekahum }. Then, **preprocessing operation** by **converting all images to grayscale** to reduce the computational complexity while keeping the important details of the images. Also, **we normalized the images** to a range of [0-255] to make sure of the uniformity of the brightness and contrast across the dataset. After this step to ensure the standard image size for all images in the dataset, we **resized the images** to a fixed resolution of 256 x 256 pixels. Then, **loading it into a dictionary whose keys are the users' IDs.**

```
user_images.keys()
```

```
dict_keys(['user065', 'user047', 'user073', 'user015', 'user051', 'user039', 'user055', 'user033', 'user027', 'user076', 'user043', 'user034', 'user071', 'user011', 'user022', 'user005', 'user082', 'user026', 'user004', 'user049',
 'user031', 'user017', 'user019', 'user057', 'user060', 'user032', 'user007', 'user075', 'user040', 'user048', 'user070', 'user016', 'user013', 'user069', 'user002', 'user045', 'user001', 'user038', 'user079', 'user030', 'user041',
 'user072', 'user010', 'user006', 'user025', 'user037', 'user063', 'user020', 'user058', 'user023', 'user018', 'user074', 'user059', 'user046', 'user021', 'user054', 'user078', 'user012', 'user029', 'user080', 'user035', 'user009',
 'user052', 'user066', 'user042', 'user050', 'user014', 'user068', 'user036', 'user056', 'user067', 'user024', 'user064', 'user044', 'user008', 'user053', 'user077', 'user061', 'user003', 'user062', 'user028', 'user081'])
```

Figure 2: loading the dictionary keys to ensure the dataset is ready to be processed

Then, **checking if there is any missing dataset** which we found 6 missing images for user48 and 50 missing ones for user59, **getting unique words in the dataset** for each user to ensure its only 10 unique words per user, **checking the missing words and handling the missing data** for user48 by duplication 10 words and using them instead of the missing ones and for the user59 by dropping it since there are no writings for 5 words so we can't duplicate from another word for the same user or from the same word for another user, **and lastly ensuring everything is working smoothly** by printing the dictionary as shown below:

```
Word counts for user048 (100 images):


abjadiyah: 10
azan: 10
fasayakfeekahum: 10
ghaleez: 10
ghazaal: 10
mehras: 10
mustadhafeen: 10
qashtah: 10
sakhar: 10
shateerah: 10
```

Figure 4: user48 after handling missing data by duplication

```
Unique words for user001 (10 total):
abjadiyah
azan
fasayakfeekahum
ghaleez
ghazaal
mehras
mustadhafeen
qashtah
sakhar
shateerah
```

Figure 3: unique words for all users, user001 for example

3. Splitting dataset train test

The dataset was separated to two groups, 80% was used to train the classifiers models and 20% was used to test the models as shown in figure [5].

```
Number of users in train set: 81
Number of users in test set: 81
User user065: 80 train images, 20 test images.
```

Figure 5: testing dataset to training and testing

4. Adding data variations through Data Augmentation

The augmentation type done to our dataset consists of adding multiple degrees of random noise {10,30,50}, and rotation by {45,30, -45, -30}. Also, doing some affine transformations which will destroy some parts of the image and lastly scaling the image by 4 scales {50%,80%,120%,150%}.and lastly, doing some illumination change by changing gamma from this set {0.5,2,3} As shown in figure [6].

Figure 6: original image vs augmented images with different types of augmentation

5. SIFT and ORB as Feature Extractor applied on original training data

Starting off with sift, the cv2 library from OpenCV is utilized to extract features from the original training data. Specifically, it uses the method **cv2.SIFT_create() with parameters** contrastThreshold = 0.02 to detect the areas with high variations and filter out the low contrast regions and edgeThreshold 5 to reduce sensitivity to edges and enhance the robustness of feature extraction. Moving to ORB its implementation was also configured from the library from OpenCV where it utilizes the **cv2.ORBcreate()** method with specific parameters, including nfeatures=1000, to detect up to 1000 robust features per image, ensuring a good balance between performance and computational efficiency. For both algorithms in order to maximize efficiency. Its output includes detailed metrics such as the total number of key points detected, descriptors for each key point grouped for each user, and the time taken for feature extraction. The results of training data on them will be discussed in table1 in the next section.

6. SIFT and ORB as Feature Extractor applied on augmented training data

For this part, the training data has been through multiple types of data augmentation such noise addition, affine transformation, scaling and rotation in multiple degrees as mentioned in point 4. And the results are in table 2 in the next section.

7. SIFT and ORB as Feature Extractor applied on raw testing data

Now sift and orb were applied to the 20% of the data which was considered for testing to extract their features and create the bag of words later on. The results are in table [4] in the next section.

8. Testing robustness for orb and sift towards variations: comparing the number of keypoints and time for each type of augmentation

In this part, the keypoints has been calculated on the original data by both sift and orb and then on each type of the augmentation types and compared to the original data as shown in table [5].

9. Training SVM models on ORB and SIFT feature vectors with and without data Augmentation and Testing their Robustness over various variations.

For these parts, the descriptors extracted by SIFT and ORB are used and KMeans is applied on them in order to create the bag of words which are the histograms that saves the occurrences of each unique feature after clustering the similar ones together as figure [1] above.

This step consisted of testing SVM model which was built on raw training data consisting of SIFT extracted features and ORB extracted features then doing exhaustive search over the best hyper parameters to select the best fine-tuning ones. then the model was tested among all testing images from the original to the different kinds of augmentation to test the model's robustness. Then the next step was that the models were once again trained on the augmented data and given the testing data and the accuracy was found on each type of augmentation to calculate robustness. All of the results are shown and discussed in table [6] and table [7] in the next section.

Results Analysis and Visualization and Comparing between approaches

ORB vs SIFT Extracting Features from Original Training Data:

Metric	SIFT	ORB
Total Keypoints Across All Users	324,437	2,114,012
Total Time for Feature Extraction	411.1127 seconds	26.3756 seconds
Average Keypoints Per User	4,005.40	26,098.91
Average Time Per User	5.0755 seconds	0.3256 seconds

Table 1: ORB vs SIFT detected key points and time taken on training data

ORB detected more key points than sift which tells that ORB is more precise than sift when detecting features among our dataset as shown in yellow. Moreover, ORB is faster than SIFT in processing each user by 15 times per user as shown in green.

ORB vs SIFT Extracting Features from Augmented Training Data:

Metric	SIFT	ORB	Observation
Total Keypoints	3,464,303	23,432,482	ORB detected **6.77 times more keypoints** than SIFT, highlighting its higher sensitivity.
Total Time (s)	4,762.16	268.90	SIFT took **17.7 times longer** than ORB to process all users, indicating higher computation time.
Avg. Keypoints/User	42,769.17	289,289.90	ORB detected, on average, **6.77 times more keypoints** per user compared to SIFT.
Avg. Time/User (s)	58.79	3.32	SIFT required **17.7 times more time per user**, making ORB far more time-efficient.

Table 2: ORB and SIFT keypoints and time taken on augmented data

Comparison On Number of Detected Key Points Between SIFT And ORB Before and After Data Augmentation for all types

Metric	SIFT Improvement Factor	ORB Improvement Factor
Total Keypoints	10.68x	11.08x
Total Time (s)	11.58x	10.19x
Avg. Keypoints/User	10.68x	11.08x
Avg. Time/User (s)	11.58x	10.19x

Table 3: ORB and SIFT improvement Factor when training on augmented data

Table [3] clearly shows that **ORB** outperforms **SIFT** in terms of improvement across all metrics, with slightly higher improvement factors for **keypoints** and **time efficiency**.

ORB vs SIFT extracting features from original Testing Data:

Metric	SIFT	ORB	Observation
Total Keypoints	82,091	533,719	ORB detected **6.5 times** more keypoints than SIFT, showing higher sensitivity.
Total Time (s)	94.3036s	6.7282s	SIFT took **14 times longer** than ORB, indicating significantly higher computation time.
Avg. Keypoints/User	1,013.47	6,589.12	ORB detected, on average, **6.5 times** more keypoints per user compared to SIFT.
Avg. Time/User (s)	1.1642s	0.0831s	SIFT required **14 times** more time per user, showing that ORB is far more time-efficient.

Table 4: ORB and SIFT keypoints and execution time on testing data

Testing Robustness for ORB and SIFT towards Variations and Comparing the Number of keypoints and time for Each Type of Augmentation:

Augmentation	SIFT - Average Keypoints Per Image	SIFT - Average Time Per Image (s)	ORB - Average Keypoints Per Image	ORB - Average Time Per Image (s)
Original	1015.98	2.3140	6606.28	0.0800
rotation_-45	962.26	2.4451	6698.84	0.0854
rotation_45	958.60	2.3526	6712.15	0.0804
rotation_-30	960.60	2.3492	6961.80	0.0822
rotation_30	956.31	2.3272	7014.36	0.0818
noise_10	1383.09	2.3848	8189.25	0.0978
noise_30	1165.86	2.3625	8085.05	0.0901
noise_50	1018.43	2.6021	7966.51	0.0871
scaling_50	692.62	2.6059	6356.33	0.0710
scaling_80	943.11	2.3327	6900.58	0.0773
scaling_120	867.19	2.2504	5474.93	0.0671
illumination_gamma_0.5	767.47	2.2405	1127.48	0.0485
illumination_gamma_2.0	1017.90	2.4103	5326.17	0.0740
illumination_gamma_3.0	1022.36	2.3266	6173.09	0.0745

Table 5: ORB and SIFT keypoints detected and time executed for each type of augmentation

It's noticed from table [5] that sift is very robust towards rotation since the number of key points detected after rotation are close to the original one. While it detected a smaller number of key points on scaling especially when it was 50% as shown in blue, but good in the 80%. For the illumination affection, it showed high of key points detected as shown in pink. On the other hand, in ORB feature extractor model, the original detected 6606 key point as shown in green and all the other augmented data showed really close numbers except the illumination with gamma =0.5 shown in red. For the time it was not that different between original and augmented data.

Testing Accuracy of SVM classifier trained on ORB and SIFT with Hyper parameter Tuning

Model	Test Accuracy	Best Accuracy (with Grid Search)
1. SIFT with SVM (no data augmentation)	34.81% Precision: 0.39 Recall: 0.35	37.35% : 1500 clusters, C=10, γ=scale, tol=0.001, poly=3, cache=200
2. ORB with SVM (no data augmentation)	37.28% Precision: 0.38, Recall: 0.37	37.53% : 1500 clusters, C=10, γ=scale, tol=0.001, poly=3, cache=200 *shown in colab notebook

Table 6: Evaluation of SVM classifier trained on augmented data extracted from both SIFT and ORB feature extractors + Grid Search to achieve best accuracy

Testing Accuracy of SVM classifier trained on Raw (Original) and Augmented Training Data Extracted from ORB and SIFT

Here is the table combining the results of SIFT and ORB on both raw training data and augmented data:

Test Data Type	SVM on SIFT (Raw Training Data)	SVM on SIFT (Augmented Trained Data)	SVM on ORB (Training Data)	SVM on ORB (Augmented Data)
Raw Test Data	34.81%	42.72%	37.28%	44.88%
Rotation Test Data	31.46%	42.43%	33.33%	43.98%
Noise Test Data	5.86%	12.47%	6.54%	20.00%
Scaling Test Data	26.69%	36.85%	16.91%	29.48%
Log-Adjusted Test Data	23.15%	32.65%	9.14%	16.73%
Affine Test Data	21.37%	31.05%	23.89%	33.70%

Table 7: Evaluation of SVM classifier trained on augmented data extracted from both SIFT and ORB feature extractor

From table [7], the accuracy was calculated for each augmentation type within its all variations grouped together as shown in each row. It's noticed that for SVM trained on SIFT extracted features the model was robust on the rotation, scaling as shown in green since they are the closest value of the testing accuracy in blue. It was weak on the noise and affine transformed data while showed moderate performance for illumination with the log adjusted test data. However, when the augmented data was utilized to build the model, the accuracies went higher but stayed in the same order.

On the other hand, for the SVM model built on ORB features the highest obtained accuracy was when testing the original data as shown in pink. The model showed robustness in rotation and affine transformation as shown in yellow but it was weak towards the noise and the log adjusted data that affected the illumination. However, when the augmented data was utilized to build the model, the accuracies went higher but stayed in the same order.

Extra Work done in the notebook

Before deciding to use the SVM machine classifier, we have tried the classifiers MLP and random forest but they gave very small accuracies less than 20% so we considered the SVM as the selected classifier and did hyper parameter tuning through grid search to find the best ones as shown in the notebook. Also, we have done a model that recognizes the word itself as a first approach before doing the second approach which is recognizing the handwritings which is shown at the end of the notebook.

Conclusion

To conclude, SIFT feature extractor model shows robustness towards scaling and rotation as noticed in our experiment while ORB feature extractor model shows robustness over rotation and affine transformations. ORB extracts feature in a faster way due to its components while SIFT needs much more time to extract keypoints as proved above. SIFT is known for its higher accuracies in most applications but in this exact application and dataset ORB showed a little bit higher accuracy. It has been noticed how increasing the data variations can lead up to better performances since the classifier gets trained on more patterns especially when the dataset is originally small as in our AWHP dataset. Lastly, we have built a strong and robust model using ORB and SIFT but the accuracies didn't cross 45% because of how small the dataset was, one solution we propose as a future work is to extend the dataset for each user so that our trained models get better with higher classifications performed correctly.