Computer Vision: Action Recognition

UP927887

Abstract— This report explores how to implement action recognition through computer vision as it's one of the main core computing concepts in computer science. The report aims to utilize modern computer vision libraires (mainly OpenCV) and takes advantage of packages such as SIFT, ORB feature detectors and extractors. This report achieves a maximum 75% accuracy score (using KFold cross-validation) with the model that uses SIFT and SVM algorithms.

Index Terms—Clustering, Face and Gesture Recognition, Feature Extraction or Construction, Feature Representation

1 Introduction

COMPUTER Vision is an essential part of core computing concepts. In order for computers to understand the real world, we have to teach the computer how to process different objects through model training and testing. In this report, the problem is the challenge of teaching computers how to recognise human actions through video captured on a camera. Therefore, action recognition will be implemented in this report using basic functionality and advanced functionality. The basic functionalities are feature representation, feature detection/extraction, data training and recognition using multiple classifiers, and analysing the experimental results using K-Fold cross-validation. The advanced functionalities expand on the basic functionalities which are using different features, using different classifiers and analysing different datasets. The approach will use Python as its programming language because of multiple packages suited for feature extraction and model training.

The report will utilize two datasets both based on human actions. The first dataset is named the "Recognition of Human Actions" made by the KTH Royal Institute of Technology [1]. It's a dataset containing six types of human actions captured on video which are: walking, jogging, running, boxing, hand waving and hand clapping.

The second dataset is named "HMDB: A Large Human Motion Database" [2], which is also a large video dataset taken from various sources such as YouTube videos or movies and it has 51 different action categories each containing a minimum of 101 videos.

In this report, the first dataset will be referred to as KTH whereas the second dataset will be referred to as HMDB.

2 Approach

2.1 Motion History Image

Python 3.11 was used in the report in order to use the latest version of OpenCV (4.7.0). OpenCV is a package available in C++, Java, JavaScript and Python. While C++ is a very powerful language to use with OpenCV, we would have to use an older version of OpenCV which would prove difficult if we wanted to use newer features.

Following this, feature extraction algorithms require still images to function properly. Therefore, Motion History Images (MHI) were implemented to convert the videos in the dataset into images. This was done by iterating through each frame in the video and using background subtraction achieved by using the NumPy package to generate a still image suitable for feature extraction. The generated MHI would then be saved into the project directory to use later in the program.

2.2 Labels

After creating and saving the MHIs into the project directory, labels representing the human action would need to be generated in order to later classify and train the model. Labels are made by reading the folder names inside the dataset and assigning each name to a dictionary as well as storing the images into a list which is later used for feature extraction.

2.3 Scale-Invariant Feature Transform (SIFT)

After generating MHIs and creating labels for the images, one of the basic functionalities is to implement feature detection and extraction. Scale-Invariant Feature Transform (SIFT) is a well-known algorithm in the computer vision world. This algorithm was used for its accuracy and robustness in feature extraction that has been tried and tested in multiple computer vision projects making it a reliable algorithm.

SIFT was implemented in the project using a function named "extractFeatures". This function takes in a list of MHI images generated earlier in the program and as it's already in grayscale, there is no longer a need to repeat the process. The program then iterates through the list and uses "sift.detectAndCompute()" to extract the descriptors (in the form of NumPy arrays) to add to a new list which is called later in the program.

2.4 Oriented FAST and Rotated BRIEF (ORB)

The intended approach to the advanced functionality portion to feature extraction was to implement Speeded Up Robust Features (SURF) [3]. However, the algorithm is still patented as of May 2023 which lead to another algorithm being used. Oriented FAST and Rotated BRIEF (ORB) was implemented as an efficient alternative to SURF, it was coded in the same function as SIFT to make switching algorithms easier during testing.

2.5 KMeans Clustering

After feature extraction, KMeans clustering is used to find patterns within the descriptors and return the clusters as a list. The reason why KMeans is used is that it's computationally efficient in model training and its scalability can handle large datasets such as possibly thousands of generated descriptors.

2.6 Support Vector Machine (SVM)

As part of basic functionality, after generating clusters from the KMeans function, the approach was to use the Support Vector Machine algorithm (SVM) to train the model. The program splits the descriptors and the labels to appropriate train and test splits using the KFold cross-validation function (An algorithm which is also part of the basic functionalities). The whole dataset gets split into five partitions and gets trained using the SVM. The average accuracy would be given after training.

2.7 Neural Network Classification

As part of the advanced functionalities, the Neural Network Multi-Layer Perceptron Classifier (MLPClassifier) was seen as another approach towards model training. The implementation of this algorithm is similar to the SVM, meaning the KFold function was also used in conjunction with the MLPClassifier to give the average accuracy the model achieves when predicting the test set.

3 Results

Using the techniques introduced in the earlier section, the results section will be split into two sections representing the two datasets used in the project. As well as outputs that were given by generating the MHI and detecting key points from SIFT and ORB, the program recorded the time taken for each function to run using the time package and it also recorded the accuracy of the model which is the average of 5 KFold "accuracy_score()" results.

3.1 The KTH Dataset

As the KTH dataset was relatively smaller than other computer vision datasets, all six classes of the dataset were able to be processed through the program without running the risk of the computer running out of memory. Because the content of the dataset already has static backgrounds and is already grayscale, most of the MHIs generated by the program have little to no background noise and are only focused on human action. The example taken from the generated HMIs is the first entry to the "boxing" class.

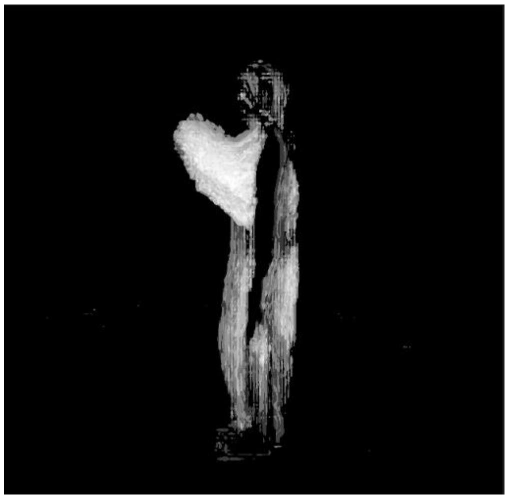

Fig. 1. Generated MHI of a person "boxing"

3.1.1 Using SVM

<u>Measuring Speed</u>

	Feature Detection/Extraction Algorithm	
Execution Time (seconds)	SIFT	ORB
MHI Processing	1456.60	1452.53
Labels and Features Generation	106.34	17.67
KMeans Cluster Generation	2708.99	200.27
Model Training	13.86	0.88
Total Time Elapsed	4285.80	1671.34

Fig. 2. Execution times for SVM

<u>Measuring Accuracy</u>

Accuracy of Feature Detection/Execution Algorithm	
SIFT	ORB
76.6%	59.59%

Fig. 3.Accuracy scores for SVM

As seen from Figure 2, the total time elapsed heavily differs between both SIFT and ORB algorithms. This is because SIFT takes 10x more time to generate the KMeans clusters than the ORB algorithm. This may be due to the nature of both feature extraction algorithms as SIFT has more detected key points than ORB (seen in Figure 4), which results in more descriptors being processed by the KMeans algorithm.

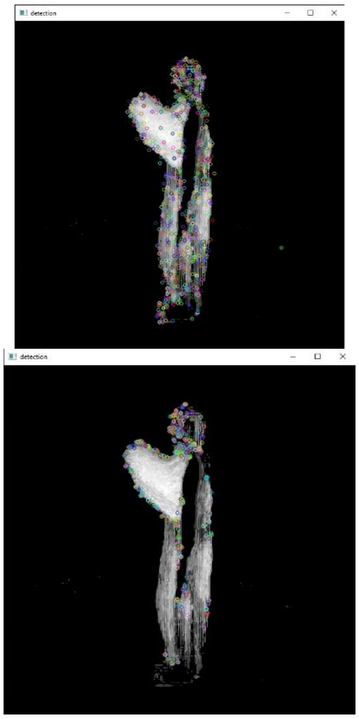

Fig. 4. Keypoints on an MHI. SIFT (Top) and ORB (Bottom)

As seen from Figure 3, it suggests that SIFT - while being slower than ORB - is more accurate than ORB, this is because SIFT features are highly descriptive and can match features across variations in scale, rotation and illumination whereas ORB prioritises speed in order to finish the model training.

3.1.2 *Using MLPClassifier*

Measuring Speed

	Feature Detection/Extraction Algorithm	
Execution Time (seconds)	SIFT	ORB
MHI Processing	1448.47	1457.75
Labels and Features Generation	106.52	16.61
KMeans Cluster Generation	2640.91	223.24
Model Training	19.47	057
Total Time Elapsed	4215.37	1703.87

Fig. 5. Execution times for MLPClassifier

As seen from Figure 5, the MLPClassifier follows a similar trend to the SVM model, where the SIFT model is slower than the ORB model. However, in Figure 6, the accuracy decreased for both the SIFT and ORB models but does not change the fact that SIFT had a higher accuracy than the ORB model.

While both SVM and MLPClassifier are robust classification models, SVMs aim to maximise the margin between different classes, meaning that they are less influenced by individual outliers. On the other hand, MLPClassifier is more sensitive to outliers because of weight updates during training, which were not explicitly specified before running the program. A scenario of the MLPClassifier being more sensitive to outliers is the similarity of different MHIs in the training/testing sets, where the "jogging" action has a similar MHI to the "running" action MHIs.

Measuring Accuracy

Accuracy of Feature Detection/Execution Algorithm	
SIFT	ORB
68.1%	57.1%

Fig. 6. Accuracy scores for MLPClassifier

3.2 The HMDB Dataset

As part of the advanced functionalities, the HMDB dataset was also explored. The initial plan for the approach was to train all 51 classes, which resulted in the computer running out of memory as it needed more than 8GB of RAM to complete the KMeans clustering. Therefore, a total of 8 classes were used for model training, which would affect the total execution time as it was a smaller subset of the original dataset.

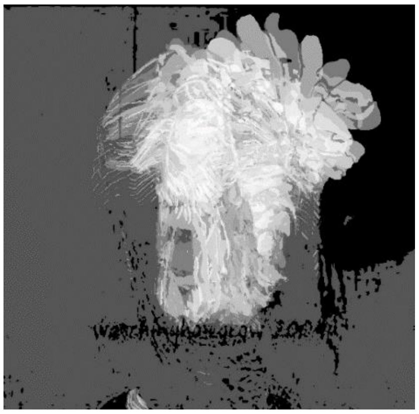

Fig. 7. Generated MHI of a person "brushing hair"

3.2.1 *Using SVM*

Measuring Speed

	Feature Detection/Extraction Algorithm	
Execution Time (seconds)	SIFT	ORB
MHI Processing	509.37	524.17
Labels and Features Generation	160.29	26.39
KMeans Cluster Generation	6071.51	330.27
Model Training	32.89	1.50
Total Time Elapsed	6774.08	882.33

Fig. 8. Execution times for MLPClassifier

As the original videos in the dataset are in colour rather than KTH's grayscale dataset, this opened the possibility of a lot of noise in the MHI image which can be seen in Figure 7 which is supposed to be a woman brushing her hair. Because of this, both SIFT and ORB models would mistake the number of descriptors in the MHIs which resulted in a longer KMeans Cluster generation. This fact would also affect the accuracy of the models from KFold cross-validation seen in Figure 10.

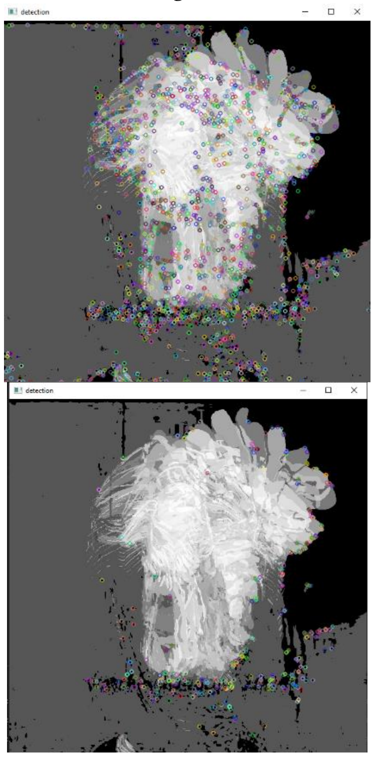

Fig. 9. Keypoints on an MHI. SIFT (Top) and ORB (Bottom)

Measuring Accuracy

Accuracy of Feature Detection/Execution Algorithm	
SIFT	ORB
47.1%	27.8%

Fig. 10. Accuracy scores for SVM

3.2.2 *Using MLPClassifier*

Measuring Speed

	Feature Detection/Extraction Algorithm	
Execution Time (seconds)	SIFT	ORB
MHI Processing	527.51	533.38
Labels and Features Generation	161.64	26.99
KMeans Cluster Generation	6226.55	344.14
Model Training	46.83	9.47
Total Time Elapsed	6962.53	913.99

Fig. 11. Execution times for MLPClassifier

Following the same trend, the SIFT and ORB accuracy rates drop when MLPClassifier is used. This supports the point that MLPClassifier is more sensitive to outliers, some of which can be seen in Figure 9. However, the behaviour of the amount of time taken to complete the program is similar to the SVM model.

Measuring Accuracy

Accuracy of Feature Detection/Execution Algorithm	
SIFT	ORB
36.9%	21.7%

Fig. 12. Accuracy scores for MLP

4 Discussion and Conclusions

4.1 Discussion of Results

Firstly, one of the main insights achieved from these experiments is SVM Classification is better suited for Action Recognition than the Neural Network MLPClassifier which can be seen by the accuracy rates in Figures 3 and 6. However, by experimenting with two separate datasets, the results suggest that in terms of execution time, there's no difference between the MLPClassifier and SVM.
Another insight taken from these experiments is that ORB is faster than SIFT, but SIFT is more accurate than ORB because of the SIFT algorithm's high distinctiveness. Therefore, for accuracy, SIFT would be the recommended approach to detect and extract features from MHIs.
Finally, by experimenting with another dataset specifically one with colours, feature detection algorithms noticeably struggle to extract features from MHIs with more noise. This would also suggest that the function that handles generating the MHIs should be improved in the next experiment to reduce more noise in order for feature extractors to function correctly.

4.2 Future Work

Expanding on the previous section, the function to generate Motion History Images could be improved by expanding on the background subtraction and introducing image filtering in order to reduce noise in the MHI. This would result in fewer outlier features detected by feature extraction algorithms and would also decrease the execution time needed to complete the program.
Another piece of improvement in this project would be to experiment with different feature detection algorithms. An example would be SURF as it's supposed to be "a faster SIFT", and after experimenting with SURF we would be able to compare if SURF is better than using SIFT. Other examples to implement in this project would be Maximally Stable Extremal Regions (MSER) or Features from Accelerated Segment Test (FAST).
Finally, the last piece of improvement that would benefit this project would be to experiment with more datasets. In this report the computer only had enough memory to experiment with a subset of HMDB, a future experiment could see all 51 classes being used or utilize another dataset such as UCF101[4], which has 101 different classes of human actions.

4.3 Conclusion

Concluding this report, the experiments proved to be a success in recognising human actions through model training and feature extraction. Judging from the results, there are two different options for optimal speed and optimal accuracy. An optimal speed model would be a model trained using SVM and using ORB as the feature detection/extraction whereas the optimal accuracy model would be a model trained using SVM and using SIFT as the feature detection/extraction algorithm. By experimenting with more features available using OpenCV or even by experimenting with packages in OpenCV's "contrib" package which has the package "xfeatures2d".

References

[1] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., Sep. 2004. doi:10.1109/icpr.2004.1334462

[2] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," 2011 International Conference on Computer Vision, Nov. 2011. doi:10.1109/iccv.2011.6126543

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," Computer Vision and Image Understanding, vol. 110, no. 3, pp. 346–359, 2008. doi:10.1016/j.cviu.2007.09.014

[4] K. Soomro, A. R. Zamir, and M. Shah, UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild, 2012. Accessed: May 25, 2023. [Online]. Available: https://www.crcv.ucf.edu/papers/UCF101_CRCV-TR-12-01.pdf