North South University

Department of Electrical and Computer Engineering
CSE445-Project Final Report
Summer-2024 | Group-05 | Section-06

Face Recognition System Using Machine Learning

SUBMITTED TO:

DR. MOHAMMAD SHIFAT-E-RABBI [MSRB]

ASSISTANT PROFESSOR

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SUBMITTED BY

Name	ID
Mir Fahad Abdullah	2121572642
Sairat Jamin Shefa	2122172042
Fatema Rahman Lamia	2021996642
Md. Raduain Hossain Rimon	2021995642

DATE OF SUBMISSION:

26TH NOVEMBER, 2024

TABLE OF CONTENTS

ABSTRACT

Emotion recognition is one of the most important applications of computer vision, enabling the system to analyze and interpret human emotions through facial expressions. The following project describes a face recognition system that classifies emotions into eight categories: Angry, Contempt, Disgust, Fear, Happy, Neutral, Sad, and Surprised. The proposed methodology involves image preprocessing, HOG feature extraction, PCA dimensionality reduction and using SVM classifier. The system to be developed is trained by using the labeled dataset and tested for accuracy and the system's resistance to noise. The obtained accuracy of the test is 91.67%, and the cross-validation results indicate the project's possibility of a scalable and efficient solution to emotion detection with future applications in different fields.

INTRODUCTION

Human emotions are usually expressed on the face and are an important aspect of human communication. The automation of expression recognition allows innovation in areas like healthcare, entertainment, education, and human-computer interaction. For example, the virtual assistant can adapt its answers to the resulting emotional state of the user, whereas the educators can utilize the emotion detection systems to assess the level of engagement among students as it happens. The aim of this project is to develop a system of emotion recognition based on facial expressions using machine learning. Contrary to the deep learning models, which need huge amounts of data and computational resources, this method unites classic machine learning techniques: HOG for the feature extraction and SVM for classification. This hybrid approach balances efficiency and accuracy, hence making it ideal for resource-constrained environments.

The primary objectives of this project are:

1. In order to control and show the practicality of the developed emotion detection system based on facial images.

2. To compare the results obtained from different classic computer vision techniques to determine their applicability in facial expressions recognition.

3. To draw attention to some of the issues and to look at how some of the approaches might be used in real situations.

BACKGROUND AND MOTIVATION

Facial expression-based emotion recognition is one of the important research topics in the computer vision field. Although deep learning has emerged as a popular form of information extraction approach, the general usage of these methods requires significant computation and data annotation. This work follows the more resource-efficient route and illustrates how traditional machine learning techniques can attain competitive results. The main motivation comes from providing a lightweight, scalable solution suitable for real-world applications. Emotion recognition has a wide range of applications in real-world scenarios; hence, it is a worthy field of research. Some of the key motivators for this project include:

- **Healthcare Applications:** Emotion recognition finds great potential in healthcare: early detection of depression or anxiety by spotting patterns in facial expressions and monitoring patients in rehabilitation or therapy by systems with this technology. Moreover, systems that are fitted with this technology could supervise the patients in rehabilitation or of physical therapy.
- **Educational Tools:** One of the differences is that an emotion-aware learning platform can change the way that the content is delivered to students because it has control over the amount of emotion involvement the students' exhibit while learning.
- **Commercial and Marketing Use Cases:** Businesses can get information about consumer moods from customers as they engage with products or ads.
- **Practicality of Classical Techniques:** Deep learning is good in many areas but it needs large databases and computation processing. This project brings out the applicability of simple techniques such as the HOG and the SVM as these are some of the most economical, explainable procedures which are quite desirable for the relatively smaller data sets.
- **Research Opportunities:** Exploring models of old school ML is beneficial in developing the holistic view of how feature extraction is done, how dimensionality reduction can be done and what classification techniques can be used.

TOOLS & RESOURCES

For this machine learning project, we will use various kinds of tools and resources such as:

Programming Language:

- **Python:** In the development of the ML models, data preprocessing and evaluation, we'll be using the Python programming language.

Libraries Used:

- **OpenCV:** When it comes to image preprocessing, we will be using OpenCV library, where restrictions are applied by converting images into greyscale levels and resize.
- **scikit-learn:** SVM classification for machine learning task, PCA for feature reduction and measure of accuracy.
- **NumPy:** We have used it for arithmetic computations and for data manipulation in the case of data having a large number of attributes.
- **Matplotlib and Seaborn:** For displaying images, results and any performance metrics that would be of importance at that given time.
- **scikit-image:** This will be used for HOG feature extraction that is important in helping to identify image structure and texture.
- **imutils:** We will use it to make the handling of datasets and files easier and effective.

Hardware:

For this project, basic standard CPU-based systems were adequate, albeit with an added feature set for GPU acceleration.

Software:

Anaconda with Jupyter Notebook: It was integrated into a single platform for coding, visualization and analysis work to support the efficiency of experimentation and development.

Dataset:

This one is the most important resource for this project. We tried to collect proper and good-quality image Datasets of different facial emotion for this project.

PROJECT DATASET

The data set consist of eighty images with ten different classes where all the classes are different emotions of humans. Every subject was photographed expressing eight different emotions where the pictures are kept in respective folders. The related, hierarchal structure makes files to be easily handled and preprocessed before they are used. Every picture is categorized depending on which emotion it portrays by using the tags from the names of the files.

To better understand the kind of data in the dataset, the images are randomly selected and shown. Part of the resulting dataset of 10 images with the respective emotion labels is shown in the next figure to illustrate the emotions and the image qualities that exist in the dataset. The visualization process involves:

Reading Images: The images are taken from the dataset directory done using OpenCV.

Color Conversion: The images are converted from BGR from Matplotlib for the correct visualization.

Label Extraction: Emotion labels are obtained from the filenames which are located in a structured folder.

Visualization: The numbers of images are shown in the single row with the labels as the titles, providing the idea of the dataset heterogeneity.

This endeavor enables the assessment of data distribution, possibility of outliers and validate if the dataset suits the project objectives well. This dataset is used to train and test the emotion recognition model that has been developed in this study.

METHODOLOGY

The methodology of the entire project encompasses four main steps: data preprocessing, feature extraction, dimensionality reduction, and model training and evaluation.

1. Data Pre-processing

Data preprocessing normalizes the data and puts it in a right form for feature extraction and classification.

1.1 Gray scaling

Images are converted to grayscale to reduce dimensions and computational cost. Although the critical information of intensity is preserved while removing the color information, which may be redundant for emotion recognition.

Convert BGR to grayscale

In [27]:
```
def colortogray(im):
    image = cv2.imread(im)  # Read image
    imgray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  # Convert to grayscale
    return imgray
```

Fig: Code of Converting BGR to Grayscale

1.2 Resizing Images

These images are resized to 128x128 pixels to standardize the input dimensions. It would help in feature extraction and provide a standard input for the machine learning model.

Resize images with a fixed size(INTER LINEAR interpolation)

In [28]:
```
#Resize the image to a fixed size
def resizeImage(image, size):
    return cv2.resize(image, (size, size), interpolation=cv2.INTER_LINEAR)  # Resize with interpolation
```

Fig: Code of Resizing Images with a Fixed Size

2. Feature Extraction

Feature extraction is a process of converting the raw image data into meaningful representations. Here, HOG is utilized-a powerful descriptor for capturing edge information.

Steps Involved in HOG Feature Extraction:

1. Converting images to grayscale.

2. Resizing images to fixed dimension.

3. Defining HOG parameters such as orientations, pixels per cell and cells per block.

4. Features extraction for each image and storing them with the corresponding labels.

Features extraction using HOG algorithm

In [29]:

```
def feature_extract(imagePaths):
    features = []  # To store HOG features
    labels = []  # To store corresponding labels

    for imagePath in imagePaths:
        # Preprocessing: Read and preprocess the image
        im = colortogray(imagePath)
        im = resizeImage(im, 128)  # Resize to 128x128 pixels

        # Extract HOG features by defining HOG parameters
        fd1 = hog(
            im,
            orientations=9,            # Number of gradient bins
            pixels_per_cell=(8, 8),    # Size of each cell in pixels
            cells_per_block=(2, 2),    # Number of cells per block
            block_norm='L2-Hys',       # Block normalization method
            transform_sqrt=False,      # No image square root transformations
            feature_vector=True,       # Return features as a single array
            visualize=False            # Do not return the visualization
        )

        # Extract label from image path
        label = imagePath.split(os.path.sep)[-2]  # Folder name represents the label
        labels.append(label)
        features.append(fd1)  # Append only the feature vector, not the visualization

    # Convert lists to NumPy arrays
    hog_features = np.array(features)
    labels = np.array(labels)
    return hog_features, labels
```

Fig: Code of Feature Extraction using HOG Algorithm

Extracting HOG features and labels

In [30]:
```
hog_features, labels = feature_extract(imagePaths)

# Display results
print("Extracting HOG features from the dataset...")
print("Feature extraction completed.")
print("The number of features per sample: " + str(hog_features.shape[1]))
print(f"Extracted features for {len(hog_features)} images.")
```

Fig: Code of Extracting HOG Features and Labels

Feature Vector Length Calculation:

- The input image size is 128×128 pixels.
- **Number of cells:**

$$\text{Cells per dimension} = \frac{\text{Image size (128)}}{\text{Pixels per cell (8)}} = 16 \text{ cells along width and height.}$$

- **Number of blocks:**

$$\text{Blocks per dimension} = \text{Cells per dimension} - (\text{Cells per block} - 1) = 16 - 1 = 15.$$

- Total blocks:

$$15 \times 15 = 225.$$

- Each block contributes $9 \times 4 = 36$ features (9 orientations, 4 cells per block).
- Total features per image:

$$225 \times 36 = 8100 \text{ features.}$$

- **The hog_features array shape is (80 x 8100)**

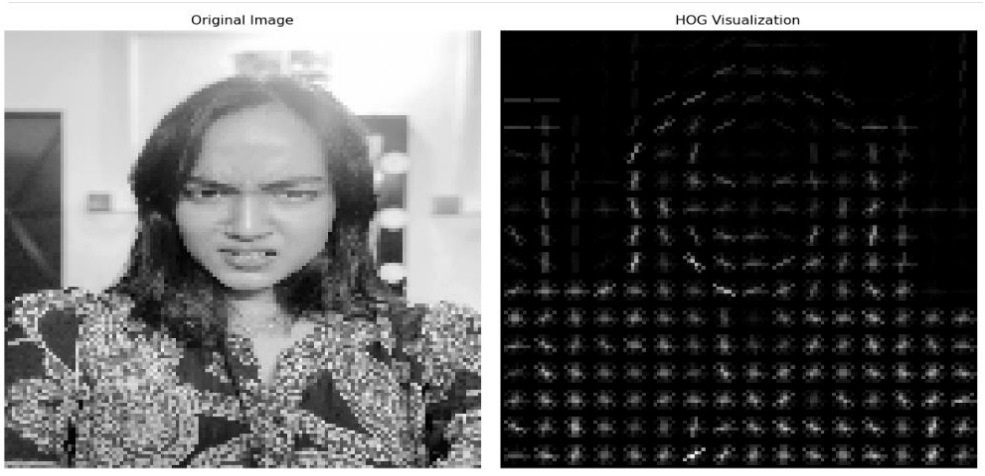

Fig: Hog Visualization

3. Dimensionality Reduction using PCA

Principal Component Analysis PCA reduces the feature space, retaining 95% of the variance as a countermeasure against the curse of dimensionality.

Steps:

(a) Scale features by standardization.

(b) Perform PCA for extracting the components that explain most of the variance.

Fig: Code of Feature Scaling

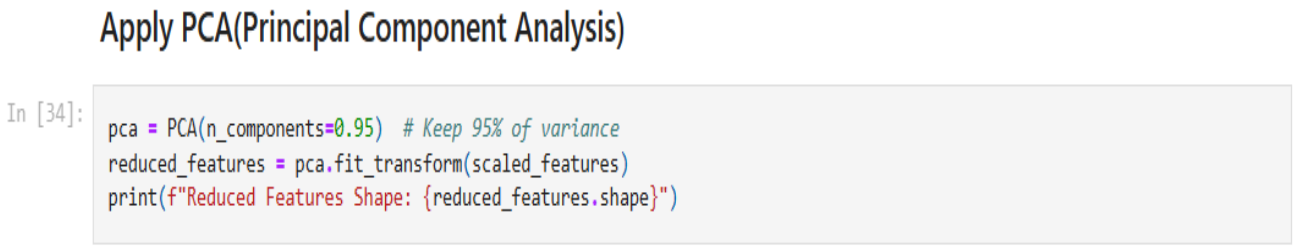

Fig: Code of Applying PCA

4. Building and Training the Model

Emotions can be classified using an SVM classifier with the RBF kernel. These steps should be taken for building and training the model:

1. Split the dataset into training (70%) and testing (30%) sets.

2. Initialize the SVM model.

3. Training the model with the training set.

Splitting dataset into training & test set

In [35]:
```
R =random.randint(1,100)
x_train, x_test, y_train, y_test = train_test_split(reduced_features, encoded_labels, test_size= 0.3,random_state=R)

print("The number of images used in training ..." + str(x_train.shape[0]))
print("The number of images used in testing ..." + str(x_test.shape[0]))
```

```
The number of images used in training ...56
The number of images used in testing ...24
```

Initialize Support Vector Classifier Model

In [36]:
```
svm_model = SVC(kernel='rbf', gamma='scale', C=1, class_weight='balanced', random_state=R)   # Radial basis function ker
```

Training SVC model

In [37]:
```
print("Training the SVM model...")
svm_model.fit(x_train, y_train)
```

Fig: Code of Splitting dataset into Train and Test Set, Initializing SVM Model, Training SVM Model

MODEL EVALUATION

Model performance is evaluated using accuracy, confusion matrix, classification report and cross validation accuracy.

Model Evaluation

In [38]:

```
# Function to calculate and display accuracy
def show_accuracy(clf, x, y, train=True):
    pred = clf.predict(x)
    data_type = "Train" if train else "Test"
    print(f"{data_type} Accuracy Score: {accuracy_score(y, pred) * 100:.2f}%")

# Function to calculate and display confusion matrix
def show_confusion_matrix(clf, x, y, train=True):
    pred = clf.predict(x)
    data_type = "Train" if train else "Test"
    print(f"{data_type} Confusion Matrix:\n\n{confusion_matrix(y, pred)}\n")

# Function to calculate and display classification report
def show_classification_report(clf, x, y, train=True):
    pred = clf.predict(x)
    data_type = "Train" if train else "Test"
    print(f"{data_type} Classification Report:\n{classification_report(y, pred)}")
```

Fig: Code of Model Evaluation

1. **Training Performance:**

Training Results

In [18]:

```
print("Training Result:\n==========================")
show_accuracy(svm_model, x_train, y_train, train=True)
print("\n")
show_confusion_matrix(svm_model, x_train, y_train, train=True)
print("\n")
show_classification_report(svm_model, x_train, y_train, train=True)
```

```
Training Result:
==========================
Train Accuracy Score: 100.00%
```

Fig: Training Result

- **Confusion Matrix**: Perfect classification across all ten classes.

```
Train Confusion Matrix:

[[7 0 0 0 0 0 0 0 0 0]
 [0 6 0 0 0 0 0 0 0 0]
 [0 0 3 0 0 0 0 0 0 0]
 [0 0 0 6 0 0 0 0 0 0]
 [0 0 0 0 4 0 0 0 0 0]
 [0 0 0 0 0 8 0 0 0 0]
 [0 0 0 0 0 0 5 0 0 0]
 [0 0 0 0 0 0 0 4 0 0]
 [0 0 0 0 0 0 0 0 6 0]
 [0 0 0 0 0 0 0 0 0 7]]
```

Fig: Trained Confusion Matrix

- **Classification Report**: Precision, Recall, and F1-score of 1.0 for all classes.

Train Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7
1	1.00	1.00	1.00	6
2	1.00	1.00	1.00	3
3	1.00	1.00	1.00	6
4	1.00	1.00	1.00	4
5	1.00	1.00	1.00	8
6	1.00	1.00	1.00	5
7	1.00	1.00	1.00	4
8	1.00	1.00	1.00	6
9	1.00	1.00	1.00	7
accuracy			1.00	56
macro avg	1.00	1.00	1.00	56
weighted avg	1.00	1.00	1.00	56

Fig: Train Classification Report

2. Testing Performance:

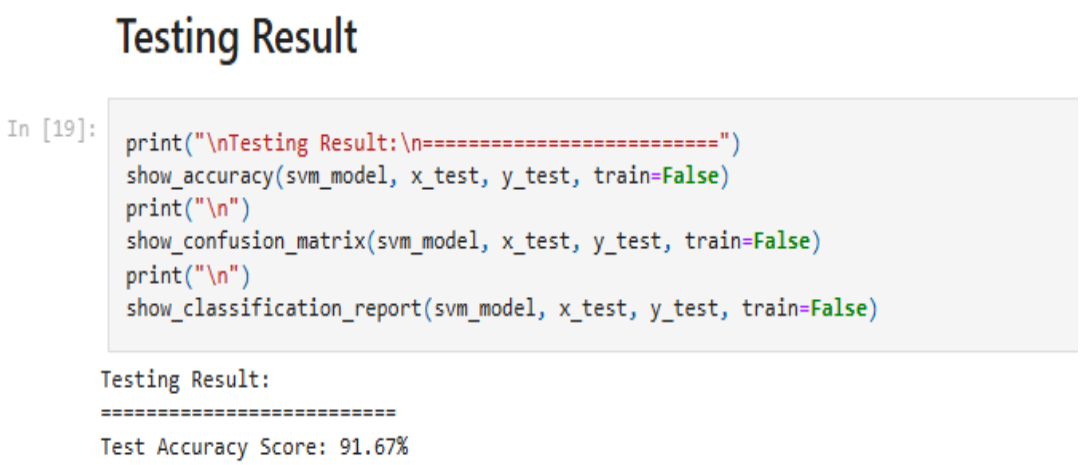

Fig: Code of Testing Result

- **Confusion Matrix**: Most emotions classified correctly but some misclassifications observed.

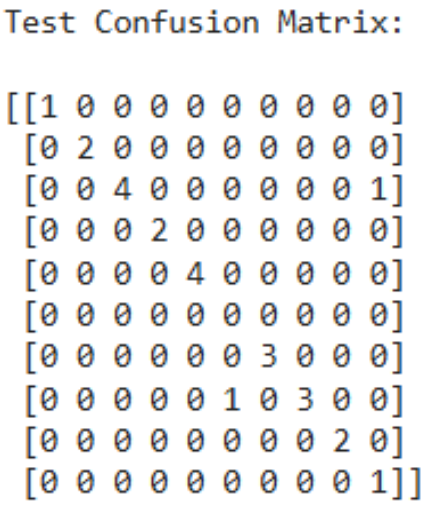

Fig: Test Confusion Matrix

- **Classification Report**: High precision and recall for most classes.

Test Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
1	1.00	1.00	1.00	2
2	1.00	0.80	0.89	5
3	1.00	1.00	1.00	2
4	1.00	1.00	1.00	4
5	0.00	0.00	0.00	0
6	1.00	1.00	1.00	3
7	1.00	0.75	0.86	4
8	1.00	1.00	1.00	2
9	0.50	1.00	0.67	1
accuracy			0.92	24
macro avg	0.85	0.86	0.84	24
weighted avg	0.98	0.92	0.94	24

Fig: Test Classification Report

3. Cross Validation:

For 3-FOLDS:

In [41]:
```
cv = KFold(n_splits=3, random_state=1, shuffle=True)
# evaluate model
scores = cross_val_score(svm_model, reduced_features, encoded_labels, scoring='accuracy', cv=cv, n_jobs=-1)
# Report performance

print('SVM MEAN  Accuracy: ',str(np.mean(scores)*100)[:5] + '%')
print('Standard deviation: ',str(np.std(scores)*100)[:5] + '%')
```

```
SVM MEAN  Accuracy:  78.82%
Standard deviation:  14.21%
```

Fig: Code of 3-FOLDS Cross Validation

For 5-FOLDS:

In [42]:
```
cv = KFold(n_splits=5, random_state=1, shuffle=True)
# evaluate model
scores = cross_val_score(svm_model, reduced_features, encoded_labels, scoring='accuracy', cv=cv, n_jobs=-1)
# Report performance

print('SVM MEAN  Accuracy: ',str(np.mean(scores)*100)[:5] + '%')
print('Standard deviation: ',str(np.std(scores)*100)[:5] + '%')
```

```
SVM MEAN  Accuracy:  97.5%
Standard deviation:  5.0%
```

Fig: Code of 5-FOLDS Cross Validation

RESULT AND DISCUSSION

Excellent performance is witnessed on training data with 100% of accuracy, whereas on testing, the data showed robust results with a remarkable accuracy of 91.67%. The confusion matrix showed some misclassifications because some emotions have similar facial expressions. In cross validation accuracy for 3-FOLDS, we got Mean Accuracy: 78.82%, Standard Deviation: 14.21% and for 5-FOLDS, we got Mean Accuracy: 97.5%, Standard Deviation: 5.0%.

Strengths:

1) Good preprocessing pipeline.
2) HOG feature representation is strong.
3) PCA could reduce dimensions without losing important information.
4) SVM did relatively well on this small data set.

Limitations:

1) Poor size of dataset impacts generalization.
2) Misclassifications in similar emotions.

FUTURE SCOPE

This paper shows that there is tremendous potential for further development and application of emotion recognition based on machine learning. While this project successfully implemented a foundational emotion recognition system, there is significant scope for future advancements:

Integration with Deep Learning: The move from decision-based engines like SVM to deep learning architectures especially CNNs improves feature learning and recognition accuracy. We could get further improvements by utilizing more sophisticated architectures as ResNet and EfficientNet.

Real-Time Emotion Recognition: Extending the system to be able pick emotions while a person is viewing videos or streaming live would come in handy in video conferencing, interactive systems and surveillance.

Multimodal Emotion Recognition: The proposal of using the facial expression in conjunction with speech, textual sentiment, and physiological information including; heart rate would result in a strong and secure emotion recognition system.

Dataset Expansion: We could enhance the capability of the model to generalize by using the dataset which is larger and includes the variation in independently and separately gathered demographics, lighting and cultural conditions.

Edge AI Implementation: Through the execution of the model in the smartphones and IoT devices, it can be very portable as well as provides offline emotion recognition model useful in remote places where there is no network connection.

Applications in Mental Health: It can be integrated with therapeutic intent, where it can be used as patient's emotion tracking, detection of existing mental diseases and current live feed during virtual therapy.

Human-Computer Interaction (HCI): This paper showed that emotion recognition could be applied in HCI systems like virtual assistants and robots so as to increase the empathy level of the system towards the user and thus improve the level of engagement of the user.

TABLE OF CONTRIBUTION

NAME	WORK
MIR FAHAD ABDULLAH **2121572642**	• FEATURE EXTRACTION (HOG'S ALGORITHM) • FEAURE SCALING AND LABEL ENCODING • DIMENSION REDUCTION USING PCA
SAIRAT JAMIN SHEFA **2122172042**	• MODELING AND LEARNING • MODEL EVALUATION
FATEMA RAHMAN LAMIA **2021996642**	• DATASET COLLECTION • CROSS VALIDATION ACCURACY CALCULATION
MD. RADUAIN HOSSAIN RIMON **2021995642**	• DATASET COLLECTION

CONCLUSION

The following paper presents a machine learning-based emotion recognition approach, which is promising for understanding and classifying human emotions through facial expressions. Tools/techniques like HOG for feature extraction, PCA for reducing the dimensionality, and SVM for classification were used to show an efficient and effective method of emotion detection.

It is an open, user-friendly system that is adaptable and thus was conceived to be integrated into a wide range of applications including healthcare, education, and human-computer interaction. Therefore, the project overcomes some of the small dataset limitations and traditional emotion analysis techniques that are critical in enabling higher accuracy and accessibility for emotion recognition systems.

In other words, this work opens the door to improving how technology can respond and learn from human emotions, leading to improved user experiences, better mental health support, and a further improvement in empathetic AI systems.

REFERENCES

[1] Neerja and E. Walia, "Face Recognition Using Improved Fast PCA Algorithm," 2008 Congress on Image and Signal Processing, Sanya, Hainan, 2008

[2] H. Ebine, Y. Shiga, M. Ikeda and O. Nakamura, "The recognition of facial expressions with automatic detection of the reference face," 2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings. Navigating to a New Era (Cat. No.00TH8492), Halifax, NS, 2000, pp. 1091-1099 vol.2.

[3] A. C. Le Ngo, Y. H. Oh, R. C. W. Phan and J. See, "Eulerian emotion magnification for subtle expression recognition," 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, 2016

[4] W. Swinkels, L. Claesen, F. Xiao and H. Shen, "SVM point-based real-time emotion detection," 2017 IEEE Conference on Dependable and Secure Computing, Taipei, 2017.

[5] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014

[6] M. Dahmane and J. Meunier, "Emotion recognition using dynamic grid-based HoG features," Face and Gesture 2011, Santa Barbara, CA, 2011

[7] C. Loconsole, C. R. Miranda, G. Augusto, A. Frisoli and V. Orvalho, "Real-time emotion recognition novel method for geometrical facial features extraction," 2014 International Conference on Computer Vision Theory and Applications (VISAPP), Lisbon, Portugal, 2014

[8] J. M. Saragih, S. Lucey and J. F. Cohn, "Real-time avatar animation from a single image," Face and Gesture 2011, Santa Barbara, CA, USA, 2011

[9] K. M. Rajesh and M. Naveenkumar, "A robust method for face recognition and face emotion detection system using support vector machines," 2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), Mysuru, 2016

[10] G. T. Kaya, "A Hybrid Model for Classification of Remote Sensing Images With Linear SVM and Support Vector Selection and Adaptation," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 6, no. 4, pp. 1988-1997, Aug. 2013

[11] J. J. Lee, M. Zia Uddin and T. S. Kim, "spatiotemporal human facial expression recognition using fisher independent component analysis and Hidden Markov Model," 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vancouver, BC, 2008

[12] X. Jiang, "A facial expression recognition model based on HMM," Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, Harbin, Heilongjiang, China, 2011

[13] Xiaoxu Zhou, Xiangsheng Huang, Bin Xu and Yangsheng Wang, "Real-time facial expression recognition based on boosted embedded hidden Markov model," Image and Graphics (ICIG'04), Third International Conference on, Hong Kong, China, 2004

[14] T. Kundu and C. Saravanan, "Advancements and recent trends in emotion recognition using facial image analysis and machine learning models," 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), Mysuru, 2017, pp. 1-6.

[15] R. Raja, ―Face detection using OpenCV and Python: A beginner‘s guide,‖ 2017.[Online] Available: https://www.superdatascience.com/opencv-face-detection/

[16] Open CV Python Tutorials. [Online] Available: http://opencv-pythontutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_table_of_contents_feature2d/p y_table_of_contents_feature2d.html

[17] D. Acevedo, P. Negri, M. E. Buemi, F. G. Fernández and M. Mejail, "A Simple Geometric-Based Descriptor for Facial Expression Recognition," 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), Washington, DC, 2017, pp. 802-808. doi: 10.1109/FG.2017.101

[18] A. Mordvintsev and K. Abid, ―Feature Detection and Description,‖ 2013. [Online]
Available:http://opencvpythontutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_ table_of_contents_feature2d/py_table_of_contents_feature2d.html

[19] P. Lucey, J. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotionspecified expression. Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010), San Francisco, USA, 94-101.

[20] A. Rosebrock, ―Detect eyes, nose, lips and jaw with dlib, OpenCV, and Python,‖ 2017. [Online] Available:
https://www.pyimagesearch.com/2017/04/10/detect-eyes-nose-lips-jawdlib-opencv-python/