

Improved Histogram of Oriented Gradient (HOG) Feature Extraction for Facial Expressions Classification

Luthfiar Ramiady, Fitri Arnia, Maulisa Oktiana*, and Andri Novandri
Department of Electrical and Computer Engineering, Universitas Syiah Kuala
Jl. Tgk. Syech Abdul Rauf No. 7, Banda Aceh 23111
*e-mail: maulisaoktiana@usk.ac.id

Abstract—Facial expression classification system is one of the implementations of machine learning (ML) that takes facial expression datasets, undergoes training, and then utilizes the trained results to recognize facial expressions in new facial images. The recognized facial expressions include anger, contempt, disgust, fear, happy, sadness, and surprise expressions. The method employed for facial feature extraction utilizes histogram-oriented gradient (HOG). This study proposes an enhancement method for HOG feature extraction by reducing the feature dimension into multiple sub-features based on gradient orientation intervals, referred to as HOG channel (HOG-C). Classifier testing techniques are divided into two methods for comparison—support vector machines (SVM) with HOG features and SVM with HOG-C features. The testing results demonstrate that SVM with HOG achieves an accuracy of 99.9% with an average training time of 18.03 minutes, while SVM with HOG-C attains a 100% accuracy with an average training time of 18.09 minutes. The testing outcomes reveal that the implementation of SVM with HOG-C successfully enhances accuracy for facial expression classification.

Keywords: *facial expression, histogram oriented gradient, machine learning, support vector machine*

I. INTRODUCTION

The facial expression recognition system, leveraging machine learning (ML) technology, has become a vital aspect in the field of artificial intelligence. With its ability to identify and analyze various human expressions such as happy, sadness, fear, anger, surprise, disgust, and neutral states, this system enables computers to comprehend human emotional expressions from facial images [1]. Its potential includes emotion recognition and more intuitive interaction between humans and machines. Real-time operation through input from a webcam/camera or video is a crucial characteristic in the development of this system, enriching human interaction with technology. The use of support vector machine (SVM) algorithms with labeled facial datasets according to expression categories is one of the methods employed [2]. Algorithm selection must consider the speed of data input processing to generate facial expression classification with minimal latency [3].

Several studies have reviewed various methods in facial expression recognition. As outlined in paper [4], this research explores the application of convolutional neural network (CNN) methods in detecting emotions in facial images. On the other hand, Yadav [5] also focuses on emotion recognition through facial expressions, utilizing SVM to recognize seven types of expressions in two individuals, such as anger, sadness, happy, disgust, neutral, fear, and surprise. Another study, discussed in [6], they combined CNN and SVM methods to reduce

model training time and enhance facial expression recognition accuracy up to 99.6%. Additionally, in [7], the facial expression recognition is classified using the Viola-Jones method. This paper delves into the use of two methods—Adaboost cascading and rectangular feature—to recognize the facial expressions. Meanwhile, in paper [8], the histogram of oriented gradient (HOG) method is employed to classify three types of facial expressions—happy, sadness, and neutral—using the JAFFA dataset.

The HOG is a technique used in image processing and object identification to extract features from images that are useful for object detection and pattern recognition [9]. The HOG focuses on identifying texture patterns and edges within images, often applied in recognizing faces, humans, or other objects. The HOG process involves several steps, including pre-processing, cell division, block normalization, histogram creation, and feature vector formation [10]. In the pre-processing stage, the image is converted into a gradient image by calculating gradient values horizontally and vertically. Cell division then divides the gradient image into small cells, followed by computing the distribution of gradient directions within each cell. Block normalization gathers cell blocks and normalizes features to reduce the effect of lighting changes. The histogram step generates a histogram for each block, illustrating the distribution of gradient directions within that block. Finally, feature vectors are formed by combining histograms from various blocks, representing the entire image. These feature vectors are then used as

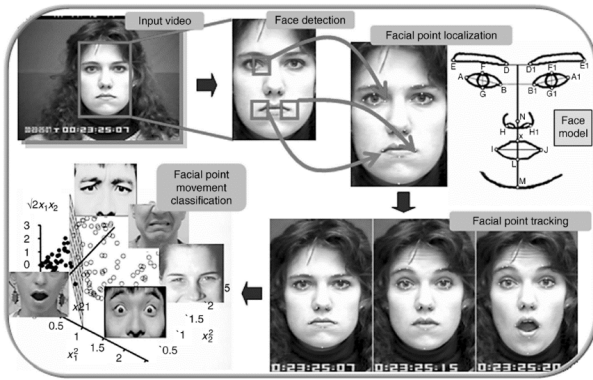


Figure 1. Facial expression recognition [16]

input in classification algorithms, such as SVM or artificial neural network (ANN) [11], [12].

Facial feature extraction is an automated recognition method of human faces by identifying their features, such as eyebrows, eyes, and lips. To enhance feature extraction in HOG, dimensionality reduction of features is performed into several sub-features based on gradient orientation intervals. Each sub-feature within each gradient orientation is known as a HOG channel. In analyzing the effectiveness of each HOG channel for each facial expression, several HOG channels are combined based on statistical similarity. The selection of HOG channels considers the best accuracy as well as optimal computational time [6]. Facial recognition systems using HOG feature extraction yield high-dimensional features. This leads to increased computational time and memory usage. The high dimensionality of HOG extraction features also impacts the input dimensions of SVM, thereby increasing computation time during training.

To address this issue, an enhancement to HOG extraction is proposed by reducing the feature dimensionality into several sub-features based on gradient orientation intervals. Each sub-feature within each gradient orientation is known as a HOG channel. The primary contribution of this paper lies in the development of a facial expression recognition system that combines improved HOG features with SVM methods to reduce computation time and memory usage.

The sections of this article are organized as follows. Section two presents the facial expression recognition system. Section three explains the proposed framework, while section four presents the simulation results and their discussion. The conclusion is provided in section five.

II. FACIAL EXPRESSION RECOGNITION

Facial expression recognition is a computer process that involves face detection to locate the image area containing only the face, known as face detection. Then, it goes through the process of facial feature extraction, such as detecting facial shapes, facial expressions, facial textures, and analyzing the detected facial feature movements, like changes in facial expressions, changes in lip shapes during smiling or sadness, changes in facial expressions when surprised or angry. This process is referred to as facial

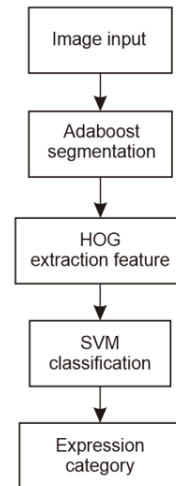


Figure 2. The proposed framework

expression interpretation [13]–[15].

The problem of human facial expression recognition can be divided into three main sub-problems, as illustrated in Figure 1 [16]:

1. The first sub-problem involves detecting human faces from various angles, where other objects might be present. This task involves identifying and marking the location of faces in the image.
2. The second sub-problem is to extract specific features from the previously detected facial regions. These features can include texture, shape, and the position of facial components like eyes, nose, and mouth.
3. The third sub-problem involves analyzing changes in the movement or appearance of facial features. This can encompass changes in the position of the mouth, eyes, and other facial muscle expressions. This information is then analyzed and classified into facial expression categories, such as emotions or facial muscle movements.

Addressing these three sub-problems is crucial in developing an accurate facial expression recognition system. Overall, this approach enables a system to identify human facial expressions, including in the context of emotion recognition.

III. THE PROPOSED FRAMEWORK

The method used in this study is illustrated in Figure 2. The first stage involves segmentation using the AdaBoost algorithm, followed by facial feature extraction using the enhanced HOG algorithm. The method used to enhance HOG extraction involves reducing feature dimensions into several sub-features based on gradient orientation intervals. Each sub-feature within each gradient orientation is referred to as a HOG channel (HOG-C). By analyzing the effectiveness of each HOG-C for each facial expression, several HOG-Cs are combined and translated based on statistical similarity. Each HOG-C is selected considering the best accuracy and computational time.

After dimensionality reduction, the extraction results are used as classification input to detect facial expressions comprising anger, contempt, disgust, fear, happy, sadness, and surprise. The dimension reduction results are utilized as input for SVM training.

A. Adaboost Segmentation

Adaboost is a machine learning technique aimed at enhancing the performance of weak models by combining them into a stronger model. In the context of image segmentation, Adaboost separates or identifies objects in images by combining several simple segmentation models into one more reliable model. The Adaboost process in image processing begins with feature selection, such as pixel intensity values, texture, or other visual characteristics that can distinguish objects from the background. The next step is to select weak models based on these features. Each weak model yields slightly better results than random guessing. Adaboost assigns weights to each data sample in the training dataset, giving higher weights to samples misclassified by the weak models. This makes subsequent models more focused on challenging cases. Adaboost combines the prediction results from weak models by giving more weight to models that perform better, forming a stronger model. This process is repeated several times, with each iteration focusing more on previously misclassified samples, thereby improving the overall model performance. The resulting Adaboost model is tested on a test dataset to measure its accuracy and performance in segmentation or object recognition tasks [17], [18] Local Binary Pattern (LBP).

The Adaboost process aims to enhance the model's capability to separate facial features from the background. By combining several weak models, Adaboost can form a stronger model that provides more accurate segmentation results. This is the primary differentiator of Adaboost from other methods that rely solely on a single model.

In this work, the Adaboost algorithm is employed for segmentation to separate features within images, particularly for isolating facial features from the background. Implementing this algorithm aids in more accurately identifying facial features, enhancing the quality of segmentation, and recognizing significant facial features.

B. Histogram Oriented Gradient Channel

The HOG feature represents the distribution of gradients and edge information within an image, which is excellent at depicting gradient changes, especially an object's shape. The computation of HOG is conducted in several stages. The first stage involves gamma calibration of the input grayscale image, using the following equation [19],

$$I(x, y) = I(x, y)^{\text{gamma}} \quad (1)$$

where, $I(x, y)$ represents the input image and gamma is the gamma constant in the grayscale image. Typically, gamma

has a value of 0.5. The second stage involves finding the gradient magnitude differences within the image. The calculation of gradient magnitude is performed using the following equation,

$$|\nabla f(x, y)| = \sqrt{G_h(x, y)^2 + G_v(x, y)^2}. \quad (2)$$

Based on (1), the gradient magnitude f is obtained from the feature vector at point (x, y) . The gradient $f(x, y)$ consists of $G_h(x, y)$ representing the gradient value in the horizontal direction and $G_v(x, y)$ representing the gradient value in the vertical direction. These two values can be defined successively by the following equations,

$$\begin{cases} G_h(x, y) = f(x+1, y) - f(x-1, y) \\ G_v(x, y) = f(x, y+1) - f(x, y-1) \end{cases} \quad \forall x, y \quad (3)$$

The third stage involves the process of considering the vector direction. Based on $G_h(x, y)$ and $G_v(x, y)$, arctan can be used to obtain the angle $\theta(x, y)$ as shown in the following equation,

$$\theta(x, y) = \tan^{-1} \left(\frac{G_h(x, y)}{G_v(x, y)} \right) \quad (4)$$

In the end, this process yields two sets of matrices: gradient magnitude and gradient direction. Subsequently, the bin values are computed based on the orientation intervals of each gradient, with a range of 20° . This results in 9 vector directions representing bin channels within each cell. The numerical range is obtained by dividing the 20° orientation interval by 180° , which is half a circle. With the assistance of these vector directions, objects are represented as gradient variations dependent on the vector direction. However, the outcomes presented in the form of gradients are challenging to interpret. Therefore, it's crucial to divide the image into numerous pixels arranged in blocks. Once these blocks are defined, the overall result will be represented in the form of a histogram. This histogram consists of bin channels containing information about the distribution of gradient directions within each block. To compute the gradient histogram within each block, the following equation is used,

$$b(j) = \bigcup_{i=1}^n h(i) \quad (5)$$

where $b(j)$ is the value within block j , $h(i)$ is the gradient histogram of cell i , and n is the number of bins in each cell. Meanwhile, the $\cup$ (union) operator is used to combine a set of indexes with specific limits. Therefore, in Equation (5), it is a union of gradient histograms $h(i)$ with a limit of n . The combination of several bins is merged into one cell [20], [21].

In the final stage, the HOG features are obtained by shifting blocks to detect objects in the image. The HOG feature can be computed using the following equation,

$$HOG = \bigcup_{j=1}^m b(j) \quad (6)$$

where m is the number of histogram cells in a block. The combination of several cells is merged into one block. An

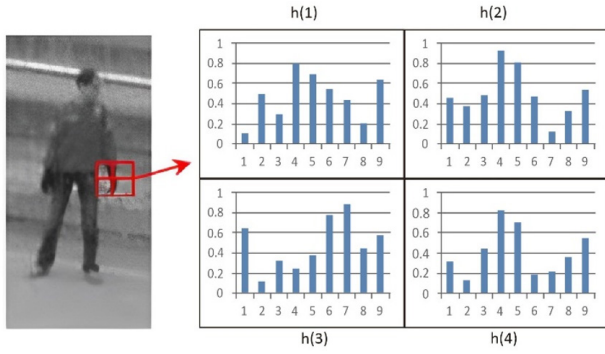


Figure 3. HOG feature extraction in a block consisting of four cells [22]

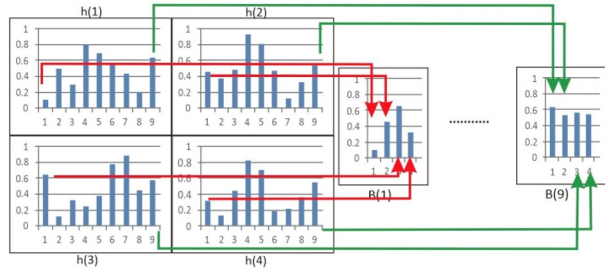


Figure 4. HOG-C feature extraction in a block [22]

example of feature extraction using HOG is illustrated in Figure 3. This feature extraction is performed by dividing the image into four histogram cells: $h(1)$, $h(2)$, $h(3)$, and $h(4)$.

C. Improved HOG-Channel Computation

This study explores updated HOG feature extraction, known as the HOG Single-Channel design or HOG-C. HOG-C is an evolution of the HOG concept, grouping multiple HOG channels based on similar gradient orientations. In HOG-C, each channel represents the same gradient orientation from pixel cells within a block. The core idea involves rearranging components with similar gradient orientations into separate channels. By organizing these channels, HOG-C produces a simpler and more efficient feature representation, reducing the total feature dimensions compared to conventional HOG [23]. However, the functionality of the conventional HOG provided by Matlab is still used to obtain gradient feature values. Original HOG feature extraction is obtained using Matlab, utilizing the function *extractHOGFeatures*(*Img*).

HOG-C is computed based on a grayscale scale with dimensions of 48×48 pixels following the previously processed dataset. Features in the original HOG are rearranged based on similar gradient orientations, thereby forming the HOG-C. The HOG-C features then form new channels consisting of 9 channels, numbered from channel 1 to channel 9, as seen in Figure 4. The process of HOG-C feature extraction is performed using (6).

$$HOG = \bigcup_{j=1}^m b(j) \quad (6)$$

The following procedure displays the refined

pseudocode for HOG-C, with variable descriptions explained in Table 1.

Procedure *HOG_Channel*(*Img*, *Cs*, *Bs*, *Nb*, *Ob*)

```

1 : Hog ← extractHOGFeatures(Img, Cs, Bs, Nb)
2 : Bpi ← floor(size(Img) / Cs - Bs) / (Bs - Ob)
3 : N ← prod(Bpi, Bs, Nb)
4 : Nib ← Bpi[1] * Bpi[2]
6 : Idx ← 1
7 : HogC ← initialize variable with N dimensions
8 : for i = 1 to Nb
9 :   for j = 1 to Nib
10 :    HogC[Idx] ← Hog[i + Nb * (j - 1)]
11 :    Idx ← Idx + 1
12 :   end for

```

Table 1. Description of variables

Variables	Descriptions
<i>Img</i>	Grayscale image data of size 48×48 pixels
<i>Cs</i>	Cell size, the size of the cell used to compute HOG
<i>Bs</i>	Block size, the size of the block used for contrast normalization in HOG
<i>Nb</i>	Number of bins, the quantity of bins in the gradient histogram within each cell
<i>Ob</i>	Overlapping blocks, the number of overlapping blocks
<i>Bpi</i>	Blocks per image, the number of blocks used in the HOG feature extraction process
<i>N</i>	Number of HOG vectors
<i>Nib</i>	Number of iterations per block, the number of iterations per block used in the iterative process for feature extraction
<i>Idx</i>	Index for loop
<i>HogC</i>	The result of the HOG-C calculation

Initially, HOG-C is extracted using the function *extractHOG-Features*(*Img*, *Cs*, *Bs*, *Nb*), a feature extraction algorithm provided by Matlab, and its output is stored in the variable *Hog*. The vector values of the *Hog* variable are generated from calculating gradient orientations on a 48×48 pixels grayscale image. A total of 19,044 feature vectors are produced through the HOG extraction process involving the subdivision of the image into small cells, computing gradient orientations in each cell, and merging these values into feature vectors for each block.

The subsequent step is to calculate the number of blocks in the image to derive the figure from the input image. Then, is computed to obtain the number of feature vectors, which is 19,044. The variable *Nib* refers to the number of blocks in the image. The algorithm executes a multilevel loop with the variable *i* in the range of 1 to *Nb* and the variable *j* in the range of 1 to *Nib*. Next, the values from the *Hog* array are mapped to the *HogC* array at the index *Idx*. The values stored in *HogC* originate from

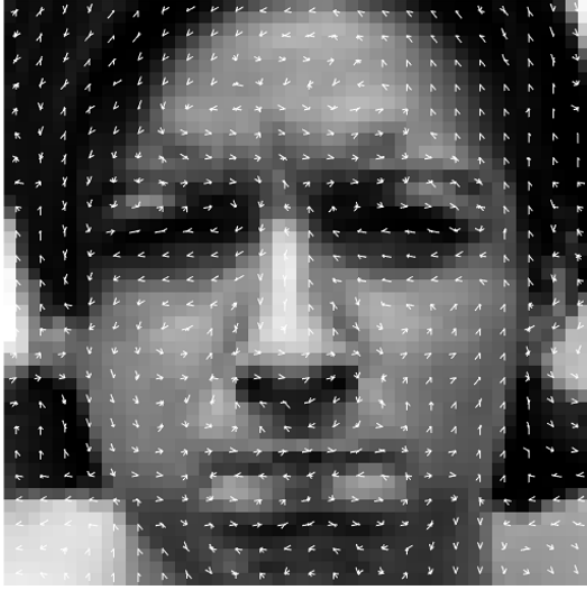


Figure 5. The sample images with angry expressions for HOG extraction

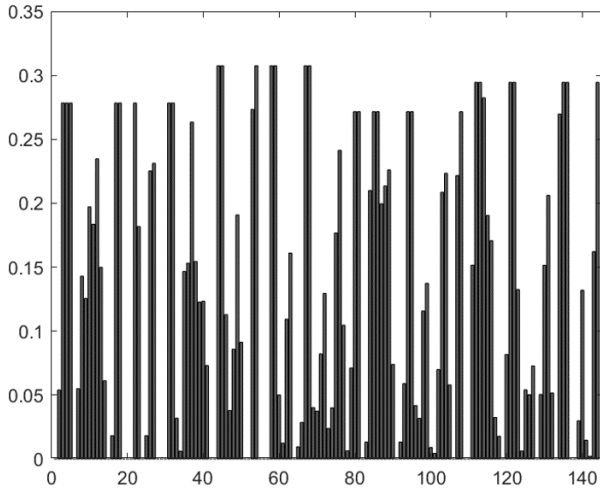


Figure 6. The results of HOG feature extraction of the initial vector value (16×9 bins)

Hog at the index $i + Nb * (j - 1)$. Each loop populates the HOG-C values by organizing each gradient based on the bin channel from 1 to 9. The result is structured vector values based on the bin channels [24].

The next step involves testing feature extraction on sample images, as depicted in Figure 5, illustrating a sample image with an angry expression. The extracted feature results are visualized with white lines on the image. After the gradient orientation feature extraction is performed, a 16×9 sample vector is taken to create a bar chart, as shown in Figure 6, representing the sample bins of 16×9 bins, capturing the initial part of the vector taken after the HOG feature extraction using Matlab. It can be observed that several gradient bin values are generated from the sample image. Following the extraction, the subsequent process involves computing HOG-C by organizing gradient features based on the bin channels from bin 1 to bin 9. The calculated HOG-C channel results can be seen in Figure 7 [25].

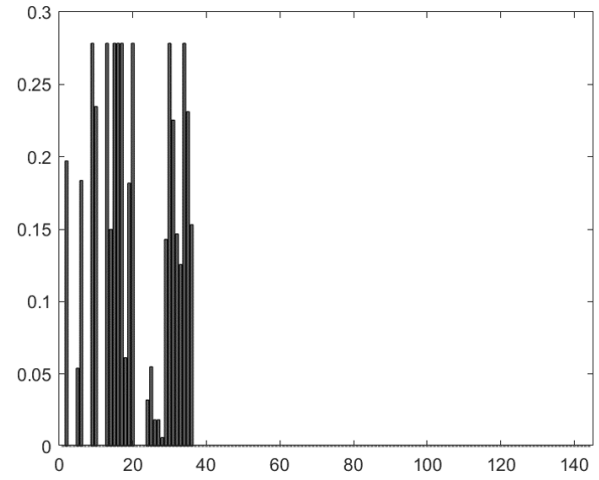


Figure 7. The results of HOG channel calculation in the initial part of the vector (16×9 bins)

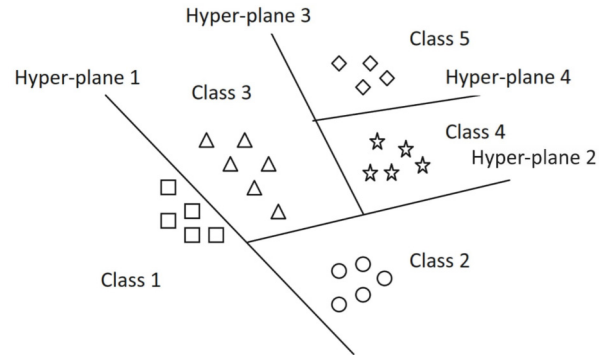


Figure 8. Multiclass SVM classification

D. Facial Expression Recognition Using Support Vector Machine

The SVM algorithm is a learning technique used for classification. SVM is fundamentally a binary classification algorithm, but there are several methods that enable the use of SVM for multiclass classification problems. Its aim is to find the optimal hyperplane that separates different classes with the maximum margin. This hyperplane works to separate each class, as depicted in Figure 8. When a data point is on one side of the hyperplane, it will be classified into one class, and if it's on the other side, it will be classified into a different class. This hyperplane can be expressed in the following equation,

$$f(x) = \text{sign}(wx + b) \quad (7)$$

where $f(x)$ is the decision function that returns the class sign based on the data position relative to the hyperplane, $\text{sign}(\cdot)$ is a sign function that returns +1 for the hyperplane if $(wx+b) > 0$ and -1 if $(wx+b) < 0$. Meanwhile, w is the weight vector, x is the feature vector, and b is the bias. The SVM training process involves finding optimal values for w and b to maximize the margin between classes. Common multiclass SVM classification methods include the one-against-rest approach and the one-against-one approach [26].

In the one-against-rest approach, an SVM model is

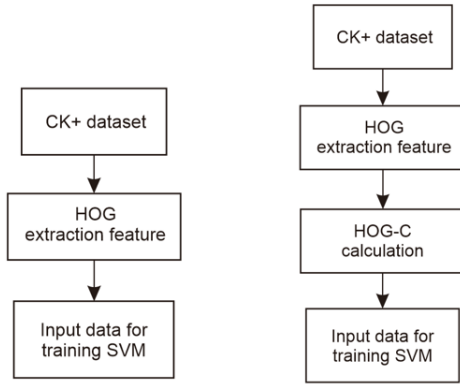


Figure 9. SVM training (a) using HOG, and (b) using HOG-C

created for each class in the dataset, aiming to separate that class from the other classes. If there are k classes, then $\frac{k(k-1)}{2}$ SVM models will be created, one for each class pair. The general equation for each SVM model can be expressed as follows,

$$\text{minimize } \frac{1}{2} \|w\| + C \sum_{i=1}^N \xi_i \quad (8)$$

with the constraint,

$$\text{minimize } \frac{1}{2} \|w\| + C \sum_{i=1}^N \xi_i \quad (8)$$

where w is the weight vector, $\phi(x_i)$ is the mapping function to the feature space, b is the bias, C is the penalty parameter controlling the trade-off between margin and the number of classification errors, and ξ_i is the slack variable allowing classification errors. During testing, each model will provide predictions. Subsequently, the class with the highest score will be chosen as the predicted class.

Meanwhile, in the one-against-one approach, an SVM model is built for each pair of classes present in the dataset. If there are k classes, then $\frac{k(k-1)}{2}$ SVM models will be created for each class pair. The general equation for each SVM model remains the same as in the one-against-rest approach. During testing, each model will provide predictions based on the class pair it handles. Then, the class with the highest score among all models will be chosen as the predicted class [27].

In the case of non-linear datasets, the “kernel trick” method is employed to map the dataset from the input space to the feature space by creating a hyperplane within the feature space. A kernel is a function that can map vectors from the input space to the feature space and compute the inner product between these vectors. Several kernel functions are commonly used in SVM, such as the polynomial kernel, radial basis function (RBF), sigmoid, and linear kernel [28].

The facial expression classification process using SVM method is performed through the Matlab application, particularly utilizing the Statistics and Machine Learning Toolbox. The SVM model design is implemented using the function `model = fitcecoc(input, label)`. *Fitcecoc* is a function to create a customizable model to select default SVM classification [29]. The relevant multiclass

Table 2. Number of imaged based on class

Class	Number of images	
	Training set	Testing set
Anger	135	10
Contempt	54	10
Disgust	177	5
Fear	75	8
Sadness	84	7
Happy	207	10
Surprise	249	10

SVM method used with the *fitcecoc* function is the Error Correcting Output Codes (ECOC). This method allows SVM to handle classification problems with more than two classes. In the ECOC method, *fitcecoc* creates a series of independent binary SVM models, each responsible for distinguishing between two specific classes from the multiclass problem. These models then collaborate to classify new samples by utilizing the output codes generated from each model.

IV. THE SIMULATION CONDITIONS

This study utilized the Cohn-Kanade+ (CK+) dataset, an extension of the CK dataset [30]. The CK+ dataset used for the training process includes a total of 593 facial expression images from 123 different subjects. Meanwhile, the dataset for testing comprises facial expression images not included in the CK+ dataset. The classification process is divided into 7 classes encompassing 7 facial expression categories. Each image has dimensions of 48×48 pixels. Details regarding the facial expression classification breakdown can be observed in Table 2.

The training process involves preparing the dataset by categorizing the image directory into 7 categories based on expressions. Each image directory is labeled according to its classification category. The Matlab program assigns data labels for each image category using the image directory names as their classes.

This research involves two methods in the training process: SVM trained using HOG features, and SVM trained using a combination of HOG-C features. In Figure 9(a), the SVM training process scheme is explained, involving the extraction of standard HOG features directly applied as input for SVM. The pseudocode for SVM training in facial expression classification using HOG is expressed as follows:

Procedure Training_Face_Classification_with_HOG

- 1 : $Cs \leftarrow 2*2 \text{ pixel}$
- 2 : $Bs \leftarrow 2*2 \text{ block}$
- 3 : $Ob \leftarrow 1*1 \text{ block}$
- 4 : $Nb \leftarrow 9 \text{ bin}$
- 6 : $Ni \leftarrow \text{size}(Ds) // \text{number of images in dataset } Ds$
- 7 : for $i = 1$ to Ni

Table 3. Number of imaged based on class

Experiment	Computation Time (s)	
	HOG	HOG-C
1	1076.04	1095.65
2	1082.62	1105.71
3	1084.83	1085.84
4	1084.03	1078.39
5	1085.05	1081.20
6	1085.22	1085.60
7	1077.53	1079.96
8	1085.24	1078.93
9	1080.06	1080.98
10	1077.45	1079.07
Mean	1081.81	1085.13

```

8 : Img ← readImage(Ds, i) //read image Ds with index i
9 : Hog[i] ← extractHOGFeatures(Img, Cs, Bs, Nb)
10 : end for
11 : Labels ← Ds.Labels //label for training
12 : ClassifierHOG ← fitcecoc(Hog, Labels)

```

Figure 9(b) depicts the training process scheme involving the extraction of standard HOG features by computing HOG-C channels before feeding them into SVM. The pseudocode for SVM training in facial expression classification using HOG-C is tabulated below:

Procedure Training_Face_Classification_with_HOG-C

```

1 : Cs ← 2*2 pixel
2 : Bs ← 2*2 block
3 : Ob ← 1*1 block
4 : Nb ← 9 bin
6 : Ni ← size(Ds) // number of images in dataset Ds
7 : for i = 1 to Ni
8 : Img ← readImage(Ds, i) //read image Ds with index i
9 : Hog[i] ← extractHOGFeatures(Img, Cs, Bs, Nb)
10 : HogC[i] ← HOG_Channel(Img, Cs, Bs, Nb)
11 : end for
12 : Labels ← Ds.Labels //label for training
13 : ClassifierHOG ← fitcecoc(HogC, Labels)

```

In the HOG-C approach, a lower-dimensional gradient vector is generated, which is then used as input for SVM. Subsequently, an analysis is conducted comparing the training results, including measuring computational time and the accuracy level of the SVM model adjusted with cross-validation on the CK+ dataset using a confusion matrix.

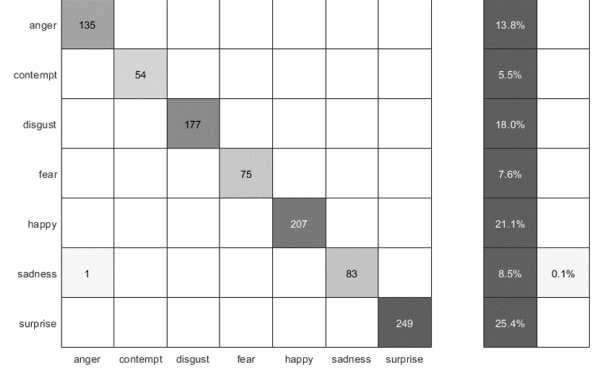


Figure 10. The results of confusion matrix HOG

Furthermore, an analysis involves comparing the test results by matching facial expression images against the obtained outcomes and calculating the negated average binary loss (NABL). NABL is an evaluation metric used in binary classification contexts. This metric is often employed to measure the model's performance in binary classification. The calculation of NABL is based on the following equation,

$$NABL = -\frac{1}{N} \left(\sum_{i=1}^N L(y_i, \hat{y}_i) \right) \quad (10)$$

where N is the total number of samples in the dataset, y_i is the actual label of the sample, $\hat{y}_i$ is the model's prediction for the sample, and L is the loss function that measures the error between y_i and $\hat{y}_i$. In general, NABL is a value that depicts the comparison of the tested model. When NABL has a positive value, it indicates that the model doesn't provide accurate predictions, and the level of loss function is relatively high.

V. RESULTS AND DISSCUSSION

This study evaluates performance based on processing time and cross-validation using a confusion matrix. The training process is repeated 10 times to obtain the average computational time. Furthermore, the computational time results from the training process using HOG feature extraction are compared to those using HOG-C feature extraction.

A. Training Analysis

In Table 3, the computation time required for the SVM training process with HOG and HOG-C feature extraction is presented. The average time for SVM training with HOG features is 1081.81 seconds, whereas for SVM with HOG-C features, it is 1085.13 seconds. The total time for training and testing, repeated 10 times, amounts to approximately 3 hours.

Next, cross-validation was performed on the CK+ dataset. The results of this cross-validation are presented in the form of a confusion matrix, as seen in Figure 10 and Figure 11. The achieved accuracy in the SVM model with

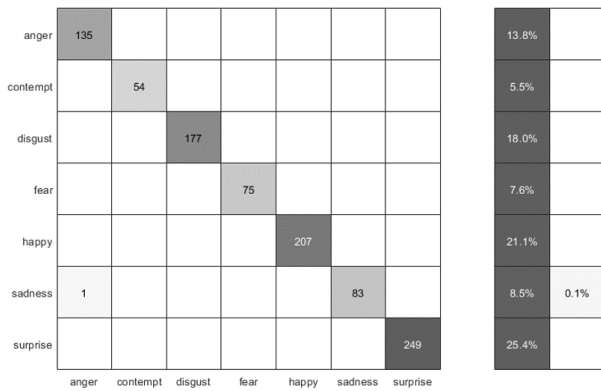


Figure 11. The results of confusion matrix HOG-C

Table 4. The comparison of computation time

No.	Description	Computation Time (s)	
		HOG	HOG-C
1.	Mean	1081.81	1085.13
2.	Median	1083.32	1081.09
3.	Std-Dev	3.69	8.91

Table 5. The comparison of accuracy

No.	Description	Accuracy (%)	
		HOG	HOG-C
1.	Anger	13.8	13.8
2.	Contempt	5.5	5.5
3.	Disgust	18.0	18.0
4.	Fear	7.6	7.6
5.	Happy	21.1	21.1
6.	Sadness	8.5	8.6
7.	Surprise	25.4	25.4
	Total	99.9	100

HOG-C is 100%, indicating that all classifications match the correct class predictions. This accuracy resulted from cross-validation on the SVM model built from the training data within the utilized dataset, with higher accuracy values signifying a better model quality.

In Table 4, the computational time comparison between HOG and HOG-C is provided, while Table 5 illustrates the accuracy comparison between the two. Although the computational time for training the classification with HOG is shorter than with HOG-C, classification using HOG demonstrates lower accuracy compared to HOG-C. The implementation of classification with HOG-C has enhanced the accuracy of the SVM model produced in this research.

The training outcomes align with the results post cross-validation, which yielded a confusion matrix showing classification using HOG achieving up to 99.9% accuracy, while classification using HOG-C achieved 100% accuracy.

The findings in [4], using the CNN method, demonstrated a facial expression classification accuracy rate of 99.33%. This validates the superiority of the

Table 6. Facial expression testing using HOG

True Class	Predicted Class	Score (NABL)	Result
Anger	Sadness	-0.0358	Not match
Contempt	Anger	0	Not match
Disgust	Anger	-0.0285	Not match
Fear	Anger	0	Not match
Happy	Anger	-0.0003	Not match
Sadness	Anger	0	Not match
Surprise	Sadness	-0.0663	Not match

Table 7. Facial expression testing using HOG -C

True Class	Predicted Class	Score (NABL)	Result
Anger	Anger	0	Match
Contempt	Sadness	-0.0224	Not match
Disgust	Disgust	-0.0182	Match
Fear	Anger	-0.0346	Not match
Sadness	Anger	-0.0467	Not match
Happy	Happy	-0.0667	Match
Surprise	Surprise	-0.0309	Match

HOG-C method in facial expression classification.

B. Testing Analysis

Next, the model was tested using facial expression images not present in the CK+ dataset. Table 6 displays the classification results using HOG, where all classifications do not match the True Class. Meanwhile, in Table 7, the facial expression classification results with HOG-C show that classifications for happy, neutral, surprise, and disgust expressions match the True Class. However, expressions of contempt, sadness, and fear show results that do not match the True Class.

VI. CONCLUSION

In this study, an enhanced HOG feature extraction called HOG-C is proposed for facial expression classification. The paper compares two different methods: the first method employs HOG, while the second method utilizes HOG-C. Both methods leverage SVM to classify facial expression categories. The simulation uses the CK+ dataset categorized into 7 expressions: anger, contempt, disgust, fear, happy, sadness, and surprise.

The results from the training process using HOG achieved an accuracy of 99.9% with an average computation time of 1081.81 seconds. Meanwhile, the results from HOG-C showed an accuracy of 100% with an average computation time of 1085.13 seconds. This indicates an additional computation time of 3.33 seconds. The added time is due to the HOG-C calculation, which requires more time. This occurs because the calculation process happens after the HOG feature extraction, resulting in an additional iteration process following the HOG feature extraction.

Even though the computation time for the HOG-C training process is longer compared to HOG, implementing classification with HOG-C can enhance the accuracy in facial expression classification. The classification test results using HOG yielded inaccurate results for all expressions, while the test using HOG-C produced accurate results for happy, anger, surprise, and disgust expressions. However, expressions of contempt, sadness, and fear did not yield accurate results.

REFERENCES

- [1] M. Pantic, "Facial expression recognition," in *Encyclopedia of Biometrics*, Boston, MA: Springer US, 2009, pp. 400–406.
- [2] Y. Ma and G. Guo, *Support Vector Machines Applications*, vol. 649, New York: Springer, 2014, pp. 23–48.
- [3] J. S. Prakash, K. A. Vignesh, C. Ashok, and R. Adithyan, "Multi class support vector machines classifier for machine vision application," in *Proc. Inter. Conf. on Machine Vision and Image Process.*, Dec. 2012, pp. 197–199.
- [4] A. Fathallah, L. Abdi, and A. Douik, "Facial expression recognition via deep learning," in *Proc. Inter. Conf. on Comp. Systems and Apps*, Oct. 2017, pp. 745–750.
- [5] S. P. Yadav, "Emotion recognition model based on facial expressions," *Multimed. Tools Appl.*, vol. 80, no. 17, pp. 26357–26379, July 2021.
- [6] L. Yang, H. Zhang, D. Li, F. Xiao, and S. Yang, "Facial expression recognition based on transfer learning and SVM," *Journal of Physics: Conference Series*, vol. 2025, no. 1, pp. 012015, Sep 2021.
- [7] K. Candra Kirana, S. Wibawanto, and H. Wahyu Herwanto, "Facial emotion recognition based on viola-jones algorithm in the learning environment," in *Proc. Inter. Seminar on Application for Technology of Information and Communication: Creative Technology for Human Life*, Sep. 2018, pp. 406–410.
- [8] P. N. Maraskolhe and A. S. Bhalchandra, "Analysis of facial expression recognition using histogram of oriented gradient (HOG)," in *Proc. Inter. Conf. on Electronics and Communication and Aerospace Technology*, Jun. 2019, pp. 1007–1011.
- [9] C. Shu, X. Ding, and C. Fang, "Histogram of the oriented gradient for face recognition," *Tsinghua Sci. Technol.*, vol. 16, no. 2, pp. 216–224, Apr 2011.
- [10] P. Kumar, S. L. Happy, and A. Routray, "A real-time robust facial expression recognition system using hog features," in *Proc. Inter. Conf. on Computing, Analytics and Security Trends*, Dec. 2016, pp. 289–293.
- [11] M. N. Chaudhari, M. Deshmukh, G. Ramrakhiani, and R. Parvatikar, "Face detection using viola jones algorithm and neural networks," in *Proc. Inter. Conf. on Computing, Communication Control and Automation*, Aug. 2018, pp. 1–6.
- [12] F. Mahmud, S. Afroge, A. Mamun, and A. Matin, "PCA and back-propagation neural network based face recognition system," in *Proc. Inter. Conf. on Computer and Information Technology*, Dec. 2015, pp. 582–587.
- [13] L. Zhang and D. Tjondronegoro, "Facial expression recognition using facial movement features," *IEEE Trans. Affect. Comput.*, vol. 2, no. 4, pp. 219–229, Oct 2011.
- [14] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Proc. Inter. Conf. on Automatic Face and Gesture Recognition*, Apr. 1998, pp. 200–205.
- [15] S. Liu, X. Tang, and D. Wang, "Facial expression recognition based on sobel operator and improved CNN-SVM," in *Proc. Inter. Conf. on Information Communication and Signal Processing*, Sep. 2020, pp. 236–240.
- [16] M. Pantic, A. Pentland, A. Nijholt, and T. S. Huang, "Human computing and machine understanding of human behavior: a survey," *Artificial Intell. Hum. Comput.*, vol. 4451, pp. 47–71, Jan 2007.
- [17] N. Kumar H N, A. S. Kumar, G. Prasad M S, and M. A. Shah, "Automatic facial expression recognition combining texture and shape features from prominent facial regions," *IET Image Process.*, vol. 17, no. 4, pp. 1111–1125, Mar 2023.
- [18] B. Islam, F. Mahmud, and A. Hossain, "High performance facial expression recognition system using facial region segmentation, fusion of HOG LBP features and multiclass SVM," in *Proc. Inter. Conf. on Electrical and Computer Engineering*, Dec. 2018, pp. 42–45.
- [19] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul 2002.
- [20] C. Tomasi. (view Feb. 2024). Histograms of oriented gradients. [Online]. Available: <https://courses.cs.duke.edu/fall17/compsci527/notes/hog.pdf>.
- [21] O. Déniz, G. Bueno, J. Salido, and F. De La Torre, "Face recognition using histograms of oriented gradients," *Pattern Recognit. Lett.*, vol. 32, no. 12, pp. 1598–1603, Sep 2011.
- [22] W. Li, H. Su, F. Pan, Q. Gao, and B. Quan, "A fast pedestrian detection via modified HOG feature," in *Proc. Inter. Conf. Chinese Control Conference*, Jul. 2015, pp. 3870–3873.
- [23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. Inter. Conf. on Computer Vision and Pattern Recognition*, Jun. 2005, vol. 1, pp. 886–893.
- [24] T. Surasak, I. Takahiro, C. H. Cheng, C. E. Wang, and P. Y. Sheng, "Histogram of oriented gradients for human detection in video," in *Proc. Inter. Conf. on Business and Industrial*, May. 2018, pp. 172–176.
- [25] S. Setumin and S. A. Suandi, "Difference of gaussian oriented gradient histogram for face sketch to photo matching," *IEEE Access*, vol. 6, pp. 39344–39352, Jul 2018.
- [26] H. I. Dino and M. B. Abdulrazzaq, "Facial expression classification based on SVM, KNN and MLP classifiers," in *Proc. Inter. Conf. on Advanced Science and Engineering*, Apr. 2019, pp. 70–75.
- [27] C. V. R. Reddy, U. S. Reddy, and K. V. K. Kishore, "Facial emotion recognition using NLPCA and SVM," *Trait. du Signal*, vol. 36, no. 1, pp. 13–22, Apr. 2019.
- [28] A. Matin, F. Mahmud, T. Ahmed, and M. S. Ejaz, "Weighted score level fusion of iris and face to identify an individual," in *Proc. Inter. Conf. on Electrical, Computer and Communication Engineering*, Feb. 2017, pp. 1–4.
- [29] S. Salimov and J. H. Yoo, "A design of small scale deep cnn model for facial expression recognition using the low resolution image datasets," *Journal Korea Inst. Electron. Commun. Sci.*, vol. 16, no. 1, pp. 75–80, Feb 2021.
- [30] P. Lucey *et al.*, "The extended cohn-kanade dataset (CK+): a complete dataset for action unit and emotion-specified expression," in *Proc. Inter. Conf. on Computer Vision and Pattern Recognition Workshops*, Jun. 2010, pp. 94–101.