Expression Recognition based on Local Directional Pattern in the Eye Region and Artificial Neural Network

Rama Patria Himawan, *TSOC, Telkom University, INDONESIA*
madokaismaiwaifu@gmail.com

Tjokorda Agung Budi W. S.T., M.T.
TSOC, Telkom University, INDONESIA

Anditya Arifianto S.T., M.T.
TSOC, Telkom University, INDONESIA

Abstract

Artificial face expression recognition is a research topic that was started ever since 90s. On this paper, writer will elaborate all the supporting algorithm used on the experiment, starting from eye-region detection using Haar cascade classifier (Haar) and Harris corres corner detection (Harris), up to expression recognition based on Local Directional Pattern (LDP) and Artificial Neural Network.(ANN)

Haar is a segmentation method based on Integral Image and classifying method named Adaptive Boosting. Haar is pretty accurate algorithm to segment an Image of face, from its background. However its performance to segment an eye-region could still be increased, hence the additional usage of Harris.

LDP Feature Extraction is an 8-directional-edge-response based feature extraction on each pixel. The final output of this process is a histogram of LDP-code on each area of the Input Image.

Artificial Neural Network is a supervised learning derived from standard linear perceptron. Especially on the usage of hidden layer, which allows the system to classify a dataset that wasn't linearly separable. On this experiment, the training phase will utilize back-propagation algorithm.

Keywords: Face recognition, local transititional pattern, contrast feature extraction, multi-layer perceptron, supervised learning, backpropagation, Haar, Harris, corner detection, eye region detection.

1 Introduction

Study of Face expression recognition is said to be done ever since Aristoteles. This research, studies how someone's feeling could be shown in his face-expression. This aspect is said to express 55% of message, as opposed to speech (7%), or Intonation (38%)[7]

There's been some researches on feature extraction based on histogram derived from convolution in a face area such as Local Phase Quantization which using Short-Term Fourier Transform, Local Binary Pattern which compares a pixel to its neighbors, and Finally Local Directional Pattern which apply Kirsrchedge Matrix to convolve an Image.

From the previous experiment, it is known, that the most expressive parts is in the eye and mouth region. Hence, on this experiment author will try to experiment on recognizing expressions, from an Image of the human eye.

2 General Overview

The flow diagram of the system could be described as the following image:

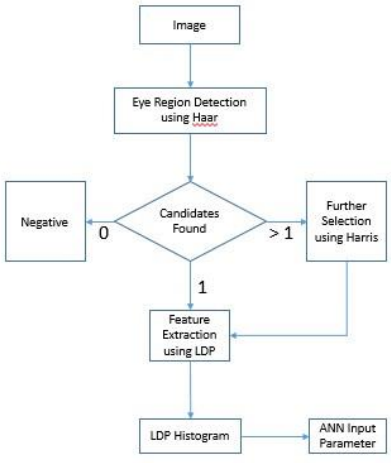


Image 2.1. System's Flow Diagram

2.1 Convolution

Convolution is an operator for a function-type operand. On a discrete function, Covolution is defined as:

$$(f * g)[n] = \sum_{m=-M}^{M} f[n-m]\,g[m]$$

(1)

In our system, convolution is the basic formula used on Harris and LDP.

2.2 Haar Cascade Classifier

Haar cascade classifier, consist of 2 main algorithms, calculation of Haar Feature using Integral Image, and Cascade classifier using Adaptive Boosting.

2.2.1 Haar Feature

Haar Feature is a feature used for image segmentation. A Haar Feature consist of 2 parts, positive part and negative part. On this experiment, both parts has the same quantities. A Haar feature is then defined as the intensity difference between the positive part and the negative part.

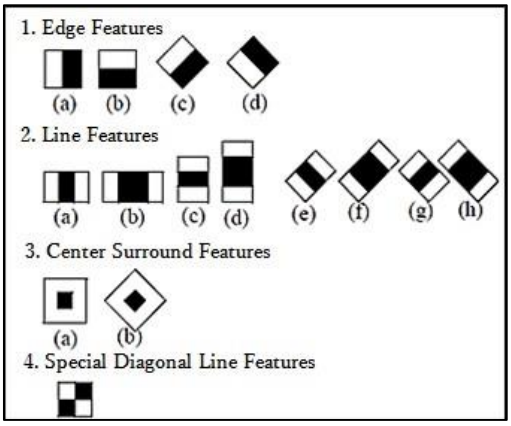


Image 2.2 Common Haar Feature used on a system

In this experiment however, author only used 4 features as shown below.

Image 2.3. Haar Feature used on this experiment

2.2.2 Integral Image

Integral Image is a dynamic programming approach to calculate the cumulative intensity of a certain area of an Image.

[000, 001, 002, 003, 004]	[000, 001, 003, 006, 010]
[005, 006, 007, 008, 009]	[005, 012, 021, 032, 045]
[010, 011, 012, 013, 014]	[015, 033, 054, 078, 105]
[015, 016, 017, 018, 019]	[030, 064, 102, 144, 190]
[020, 021, 022, 023, 024]	[050, 105, 165, 230, 300]

Image 2.4 image and its integral image

An Integral Image can be constructed by using this recursive discrete function.

$$ii[x,y] = \begin{cases} ii[x-1,y] + ii[x,y-1] - ii[x-1,y-1] + i[x,y] & x \geq 0; y \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

(2)

$$ii[x,y] = \text{integral image at } [x,y]$$
$$i[x,y] = \text{original image at } [x,y]$$

Once an integral image is constructed, the cumulative intensity at any point (x, y) to (x', y') can be calculated with:

$$\sum_{x=0}^{x'} \sum_{y=0}^{y'} i[x,y] = ii[x',y'] - ii[x-1,y'] - ii[x',y-1] + ii[x-1,y-1]$$

(3)

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Image 2.4 Intensity Matrix with marked area

0	1	3	6	10
5	12	21	32	45
15	33	54	78	105
30	64	102	144	190
50	105	165	230	300

Image 2.5 Integral Image of image 2.4 with marked area

For example in the image above, the cumulative intensity in the area colored black is equivalent to (index start from zero):

$$\text{[illegible}[4,4] - \text{[illegible}[2,1] - \text{[illegible}[4,1] + \text{[illegible}[1,1]$$

$$= 300 - 105 - 45 + 12$$
$$= 162 \quad (4)$$

2.2.3 Cascade Classifier.

Cascade classifier is a multi-stage classifier composed of AdaBoost instances, starting from the weakest to the strongest instance.

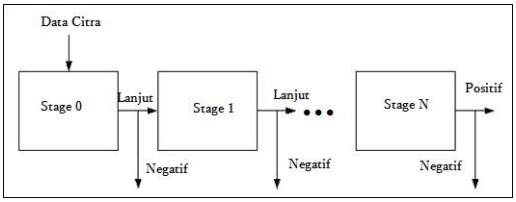


Image 2.6 Cascade classifier.

The difference between a weak adaboost instance and the stronger one is the amount of its inequality expression.

The python source code below explain what happens to an adaboost instance when a new inequality function is added as a weak classifier.

```
def addPredicate(self, predicate):
    "predicate: => Boolean"
    equation = lambda args: predicate(args) * 2 - 1
    # transform the predicate to a function that
    yield either -1 or 1
    errors = [data.cls != equation(data.args) for
        data in self.dataset]
    et = sum(data.weight * error for data, error in
        zip(self.dataset, errors))
    alpha = 0.5 * log((1 - et) / et)
    for data, error in zip(self.dataset, errors):
        data.weight = data.weight * exp(alpha) if
            error else data.weight * exp(-alpha)
    equation.alpha = alpha
    print equation.alpha
    self._separator.append(equation)
```

While the predicting function is defined as follow:

```
def predict(self, args):
    evaluated = sum(separator(args) * separator.alpha
        for separator in self._separator)
    return 1 if evaluated > 0 else -1
```

2.2.4 Harris Corner Detection

As explained in the abstract, Haar cascade classifier, while it yields a good false negative rate, it suffers from detecting false positive data, where this experiment requires the input to be a completely valid data. Thus, in this experiment the eye region detection is also supported by Harris Corner Detection to provide better result.

Harris Corner Detection, is an algorithm based on the eigenvalue of the covariant of the distribution of the derivation of the Image in regard to x-axis, and y-axis.

The derivation value can be obtained through convolution with the following matrix

$$\frac{\partial I(x,y)}{\partial x} = (I * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix})$$

$$\frac{\partial I(x,y)}{\partial y} = (I * \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}) \quad (5)$$

Having obtained the first derivation on each point in

regard to x, and y, the likeliness of a feature point existing in the current window, can be calculated by finding the eigenvalue of the covariance matrix:

Eigenvalue of a matrix is defined as:

$$\text{Eigenvalue}(A) = \det^{-1}(\det(A - \lambda I) = 0) \quad (6)$$

Where

$$A = \begin{bmatrix} \text{cov}(x_p, x_p) & \text{cov}(x_p, y_p) \\ \text{cov}(y_p, x_p) & \text{cov}(y_p, y_p) \end{bmatrix}$$

$$\text{cov}(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{n} p(x_i, y_j) * (x_i - \mu_x)(y_j - \mu_y)$$

Since A is a 2x2 Matrix, the eigenvalue, can be obtained with

$$\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$a = \sigma_x^2 * \sigma_y^2 - 2\sigma_{xy}$$

$$b = -(\sigma_x^2 + \sigma_y^2)$$

$$c = 1 \quad 2 \quad 2 \quad (7)$$

2.3 Local Directional Pattern

Local Directional Pattern or LDP, used 8 discrete function named Kirschedge Matrix as shown below:

$$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \text{East } M_0 \quad \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \text{North East } M_1 \quad \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \text{North } M_2 \quad \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \text{North West } M_3$$

$$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \text{West } M_4 \quad \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} \text{South West } M_5 \quad \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \text{South } M_6 \quad \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix} \text{South East } M_7$$

Image 2.7 Kirschedge Matrixs

By using Kirschedge Matrix, naive LDP can be described as:

$$\text{LDP}(\text{naive}) = \sum_{i=0}^{7} s(m_i \text{ naive} - m_j \text{ naive}) * 2^i; \quad s = \begin{cases} 1, & \geq 0 \\ 0, & < 0 \end{cases}$$

$$m_i(\text{naive}) = (\text{naive} * k_i)[x, y]$$

Where:

j = the matrix index that yield the k-th strongest response.

However, a faster LDP can be achieved by redefining $m_i(\text{naive})$ as:

$$m_i(\text{naive}) = \begin{cases} (\text{naive} * k_i), & i = 0 \\ m_{i-1}(\text{naive}) - 8m_{i-1}(\text{naive}) + 8m_{i-1}(\text{naive}), & i > 0 \end{cases} \quad (10)$$

Where

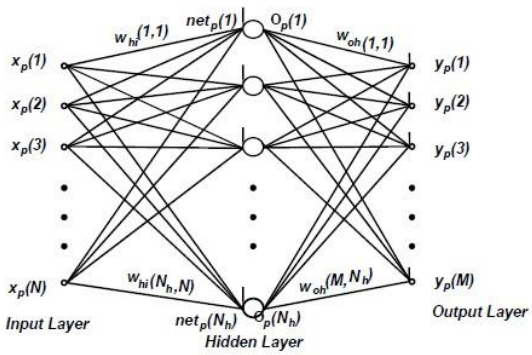
$$m_0 = (\text{naive} * k_0)[x, y] \text{ and } m_1 = (\text{naive} * k_1)[x, y] - 3$$
$$m_0 = (\text{naive} * k_0)[x, y] \text{ and } m_1 = (\text{naive} * k_1)[x, y] - 3$$

Though, this algorithm still wouldn't be as fast as convolution using Fast Fourier Transform.

To reduce the significant loss of the information about location, after the image is convoluted it is divided into an m * n area. Then, the histogram of the LDP code on each region is forwarded as an Input Parameter to the classification method namely Artificial Neural Network.

2.4 MLP (Multi-Layer Perceptron)

MLP is a feed-forward Neural Network model, consisting of n-layer. In this experiment, author used a model of 3 layers; 1 Input Layer, 1 Hidden Layer, and 1 Output Layer. As seen in the following image:



The Input layer consists of n-neuron, where n is the amount of input parameter, which is the number of distinct value an LDP of level k could achieve. The output layer consists of m-neuron where m is the amount of possible output which is 7 in this experiment, because there's 7 possible expressions. Lastly, the number of neuron in hidden layer is configurable, and is a parameter

to be experimented. While ANN could tolerate misconfiguration, too much neuron in hidden layer is analogous to having more equation than there is free variables, hence the system wouldn't be able to provide a good generalization, while too few neuron in hidden layer would likely reduce the robustness of the system.

Other than number of neuron in hidden layer, there are also other ANN parameters to be configured, namely learning rate and momentum.

Learning rate is the parameter that configures how fast a correction is applied on the system. Setting learning rate too high could lead system to stuck and failing to reach the actual minima point, while setting it too low, would possibly lead system to fail to reach convergence point because it reach the epoch limit first.

Momentum adds variance by adding a portion of previous change to system, other than the feedback correction. Having big momentum, would allow system to avoid local minima, and possibly increase the speed for system to be convergent. However having it too big could also lead system to avoid the actual global minima.

Activation function is a function that maps an input ranging from $-\infty$ to ∞ to a value between -1 and 1. Most common function used in ANN-based system is

$$\text{[illegible]}(\text{[illegible]}) = \frac{1}{1 + \text{[illegible]}}$$

(10)

However recent research shows that

$$\text{[illegible]}(\text{[illegible]}) = \tanh(\text{[illegible]}(\text{[illegible]});$$
$$\text{[illegible]}(\text{[illegible]} = \sum_{\text{[illegible]}=1}^{\text{[illegible]}=1} \text{[illegible]}(\text{[illegible]} * \text{[illegible]}(\text{[illegible]}$$

(11)

Could provide a better generalization, hence it is used in this experiment.

Where

$W_{hi}(j, k)$ = weight of the course between the j-th input node, to the k-th hidden node.

$X_n(k)$ = the activation value in the k-th input node.

2.4.1 Backpropagation

Backpropagation is a learning method used in the Artificial Neural Network to provide a correction, given a known error between an input parameter to its expected output.

Change of weighting in a node is defined as:

$$\text{[illegible]}(\text{[illegible]}) \leftarrow \text{[illegible]}(\text{[illegible]}) + \text{[illegible]} * \frac{-\delta \text{[illegible]}}{\delta \text{[illegible]}(\text{[illegible]})};$$
$$\frac{\delta \text{[illegible]}}{\delta \text{[illegible]}(\text{[illegible]})} = -\text{[illegible]}(\text{[illegible]} * \text{[illegible]}(\text{[illegible]});$$
$$\text{[illegible]}(\text{[illegible]} = -\frac{\delta \text{[illegible]}}{\delta \text{[illegible]}(\text{[illegible]}}$$

(12)

Where

$$\text{[illegible]}(\text{[illegible]}) = \text{[illegible]}$$

3. Experiment Result

Experiment is tested on JAFFE dataset consisting of 10 person showing 7 expression, including neutral. Each person show the same expression 3 to 4 times, yielding a total of 213 Image.

The Experiment consist of 2 sub-experiment:

a) The performance of eye-region detection algorithm, and

b) The performance of expression recognition system, on a varying parameter configuration. e.g. level of LDP, amount of neuron on hidden layer, the learning rate and the value of momentum.

3.1 Eye Recognition Performance

Using Harris	Actual/Expected	True	False
Non-using	True	95.3	4.7
Non-Using	False	12.2	88.8
Using	True	79.8	20.2
Using	False	0	100

Table 3.1 Performance comparison between Haris and non-Harris recognition system.

3.2.1 System accuracy on default ANN setting for each level of LDP.

K	LDP Vector Length	Averaged Accuracy
1	96	59.77%
2	336	67.81%
3	672	84.32%
4	840	86.2%

Table 3.2 Relation Table between K, LDP Vector's length, and accuracy.

As seen in the table above, the best performance obtained by k=4, however, as it doesn't increase the performance significantly compared to the k=3 which have a significantly lower vector length. Hence the value of k will be set to 3 on the following experiment.

3.2.2 System's accuracy on varying amount of hidden neuron and learning rate.

No	NH	LR	M	Akurasi (5x pengujian)
1	10	0.02	0.01	74.2%
2	10	0.05	0.01	70.2%
3	10	0.10	0.01	67.7%
4	25	0.02	0.01	84.3%
5	25	0.05	0.01	80.5%
6	25	0.10	0.01	82.2%
7	50	0.02	0.01	65.3%
8	50	0.05	0.01	66.1%
9	50	0.10	0.01	58.8%

Table 3.3 System performance on varying configuration.

4. Analysis Summary

Haar did well enough on classifying the eye-region from the non-eye-region, however, due to the constraint in this experiment, it's required for the data input to be actually correct, before submitted to the following subsequent process, thus this system also include Harris corner detection algorithm to further filter an image.

Having done the aforementioned experiment, it's known that the best accuracy can be obtained by using LDP level 4with an accuracy of 86.2%. This is because the more dimension there is, the easier for the system to separate data from one and the other. However, it inflict more drawbacks the system time consumption more than the benefit of increased accuracy as opposed to LDP level 3, hence the usage of LDP level 3.

5. Summary

1. Artificial Neural Network, provide a good result to classify expresion using LDP, with the best averaged is 84.3%.

2. System could segment eye region by using haar cascade, combined with Harris Corner Detection, that it yield accuracy of 79.8%.

3. Optimum configuration obtained with 10 neuron on hidden layer, learning rate of 0.02, and momentum of 0.01, which achieve accuracy of 84.3%.

4. FasterLDP while not as fast as FFT based convolution, could increase the speed to twice the original speed.

4.2 Suggestion

1. The eye region detection on this experiment took a significantly long, hence a better configuration on it should be done. Especially the number of stage, and the amount of rule for each stage.

2. This experiment involve a heavy computation and also doesn't rely to non-built-in package of python. This is due to the specific purpose of this experiment is for the writer to learn the underlying theory on expression recognition. However, for a production-level application, the speed could be greatly increased by utilizing the package such numpy, scipy, and sci-kit, especially the numpy.signal.convolve2d method.

Bibliography

[1] C. Szegedy, A. Toshev and D. Erhan, "Deep Neural Network for Object Detection".

[2] T. Jabid, H. Kabir and O. Chae, "Robust Facial Expression Recognition Based on Local Directional Pattern," 2010.

[3] S. B. S. Matabei, "Face Expression Recognition Using Local Directional Pattern Method and Support Vector Machine," 2014.

[4] Author, Engineering Mathematics: Open Learning Unit Level 1.

[5] P. Viola and M. Jones, "Rapid Object Detection using Boosted Cascade of Simple Features," 2001.

[6] H. Kruppa, C. M. Santana and B. Schiele, "Fast and Robust Face Finding via Local Context," 2003.

[7] J. Malik, R. Dahiya and G. Sainarayanan, "Harris Operator Corner Detection using Sliding Window Method".

[8] W. H. Delashmit, "Recent Development in MultiLayer Perceptron," 2005.

[9] Y. Lecun, L. Bottou, G. B. Orr and K. R. Muller, "Efficient Backprop," 1998.

[10] C. Angelico, "PEP - 0463 Exception-catching expressions," 2014.