BAB III
LANDASAN TEORI

3.1 Autentifikasi Ke-2

Teknologi untuk mendeteksi objek terus berkembang semakin baik, hal ini didukung dengan perkembangan ilmu pengetahuan dan komputer yang makin canggih dan lebih ramping serta tepat guna. Banyak hal yang berguna didapatkan dari informasi yang dikumpulkan pada pendeteksian objek. Salah satunya adalah kita mampu menghitung jumlah objek yang dideteksi dan dapat dimanfaatkan untuk tujuan tertentu atau keperluan lainnya.

Salah satu alat pendeteksi objek adalah menggunakan sensor kamera. Dengan memanfaatkan kamera akan ditangkap gambar atau video yang akan dianalisa dan kemudian dapat dilakukan perhitungan dari hasil berupa sejumlah objek yang dideteksi.

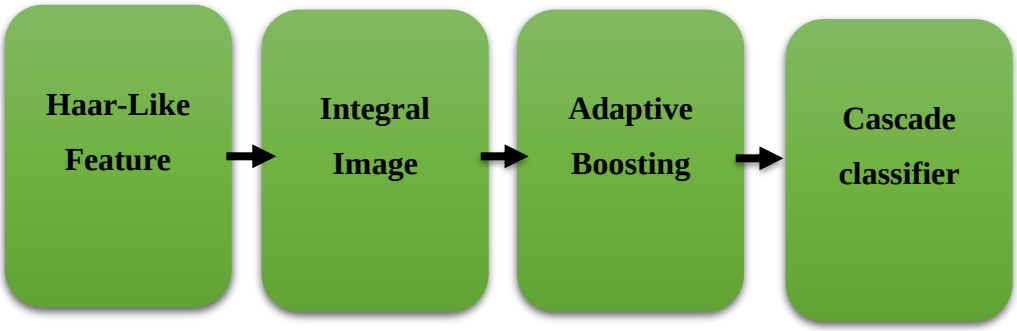
Saat ini telah banyak berkembang sistem yang memanfaatkan fitur deteksi wajah diantaranya yaitu sistem akses keamanan maupun sistem kontrol. Deteksi wajah sendiri dapat dilakukan dengan berbagai cara, salah satunya menggunakan metode Algoritma *Haar Cascade Classifier* dan *Local Binary Pattern*

3.1.1 Algoritma Haar Cascade Classifier

Algoritma Haar Cascade Classifier adalah salah satu algoritma yang digunakan untuk mendeteksi sebuah wajah. Algoritma tersebut mampu mendeteksi dengan cepat dan realtime sebuah benda termasuk wajah manusia. Algoritma Haar Cascade Classifier memiliki kelebihan yaitu perihal komputasi yang cepat karena tersebut hanya bergantung pada jumlah piksel dalam persegi dari sebuah image (Datta et al., 2015).

Pada proses pendeteksian objek dengan menggunakan metode alogritma Haar Cascade Classifier yang diusulkan oleh Viola-Jones, ada beberapa proses yang dilakukan sebelum akhirnya akan menghasilkan sebuah output objek yang terdeteksi pada sebuah citra. Dalam deteksi objek dengan metode ini, proses-proses

tersebut yaitu Haar-Like Feature, Integral image, Adaboost (Adaptive Boosting), dan Cascade Classifier(Datta et al., 2015). Skema proses dari tiap-tiap tahap yang dilalui oleh sebuah citra untuk memperoleh hasil pendeteksian objek dapat dilihat pada Gambar berikut:

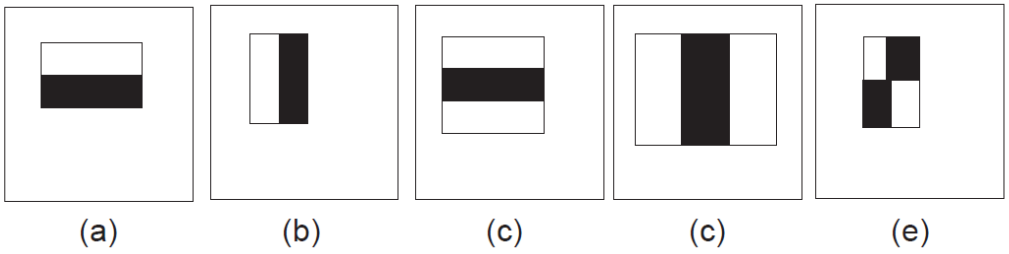


Gambar 3.1.1 Algoritma Haar Cascades Classifier

3.1.1.1 Features

Prosedur deteksi wajah Viola-Jones mengklasifikasikan gambar berdasarkan nilai fitur sederhana. Ada banyak motivasi untuk menggunakan fitur daripada piksel secara langsung. Alasan paling umum adalah bahwa fitur dapat bertindak untuk menyandikan pengetahuan domain adhoc yang sulit dipelajari menggunakan sejumlah data pelatihan. kedua untuk mengakses fitur terkait dengan gagasan bahwa sistem berbasis fitur beroperasi jauh lebih cepat daripada sistem berbasis piksel.

Algoritma Viola-Jones menggunakan fitur seperti Haar, yaitu produk skalar antara gambar dan beberapa template seperti Haar. Lebih spesifik, ia menggunakan tiga jenis fitur: 1) fitur dua-persegi panjang, 2) fitur tiga-persegi panjang dan 3) fitur empat-persegi panjang. Nilai fitur dua persegi panjang adalah perbedaan antara jumlah piksel dalam dua wilayah persegi panjang. Fitur tiga persegi panjang menghitung jumlah dalam dua persegi panjang luar dikurangi dari jumlah dalam persegi panjang tengah. kemudian, fitur empat persegi panjang menghitung antara pasangan diagonal persegi panjang. Daerah memiliki ukuran dan bentuk yang sama dan berdekatan secara horizontal atau vertikal. Lima pola Haarlike ditunjukkan pada Gambar 3.1.1.1

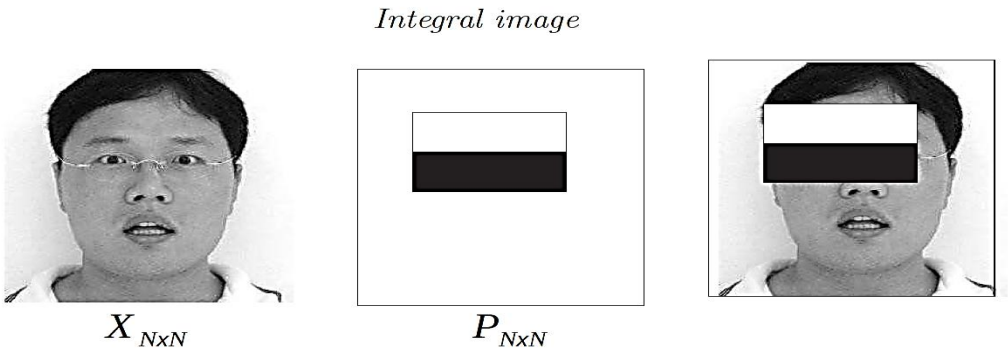


Gambar 3.1.1.1: Fitur

Gambar 3.1.1.1: Contoh fitur persegi panjang yang ditampilkan relatif terhadap jendela deteksi terlampir. Jumlah piksel yang terletak di dalam persegi panjang putih dikurangi dari jumlah piksel dalam persegi panjang abu-abu. Fitur dua-persegi panjang ditunjukkan dalam (a) dan (b), fitur tiga-persegi panjang ditunjukkan dalam (c) dan (d) dan fitur empat-persegi panjang ditunjukkan dalam (e) [1].

Ukuran dan posisi dukungan pola dapat bervariasi asalkan persegi panjang hitam dan putihnya memiliki dimensi yang sama, saling berbatasan dan menjaga posisi relatifnya. Biarkan I dan P menunjukkan gambar dan pola, keduanya berukuran sama N x N (seperti Gambar 7.2). Fitur yang terkait dengan pola P gambar X didefinisikan oleh

$$\sum_{i=1}^{N}\sum_{j=1}^{N} X(i,j)1p(i,j)white - \sum_{i=1}^{N}\sum_{j=1}^{N} X(i,j)1p(i,j)black$$



GAMBAR 3.1.3: Fitur :Haar. Hanya piksel yang ditandai hitam atau putih digunakan ketika fitur yang sesuai maka dihitung.

3.1.1.2 Integral image

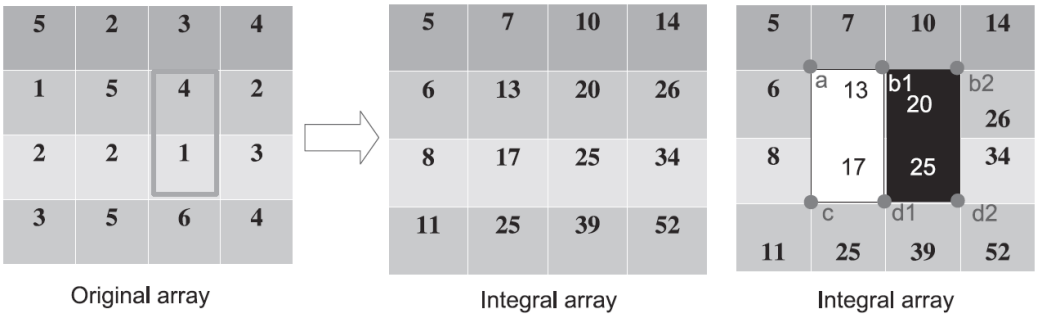
Fitur persegi panjang dapat dihitung dengan sangat cepat menggunakan representasi perantara untuk gambar yang disebut gambar integral. Gambar integral dapat dihitung dalam satu lintasan di atas gambar asli seperti yang ditunjukkan pada Gambar 7.3. Gambar integral di lokasi (i; j) berisi jumlah piksel di atas dan di sebelah kiri (i; j), dan diberikan oleh

$$II(i,j) = \sum_{i \le i`, j \le j`} I(i,j)$$

di mana II(i; j) adalah gambar integral dan I(i; j) adalah gambar asli. Menggunakan pasangan pengulangan berikut:

$$S(i,j) = S(i,j-1) + I(i,j)$$
$$II(i,j) = II(i-1,j) + s(i,j)$$

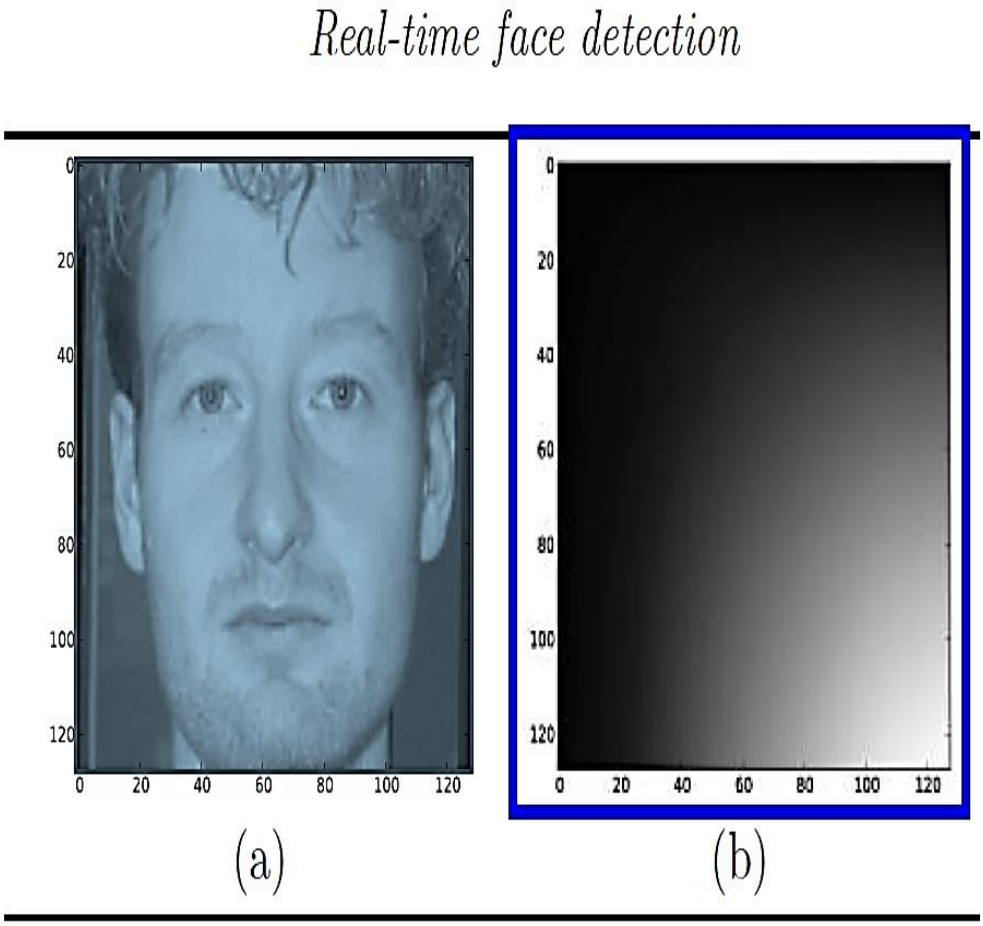
di mana S(i; j) adalah jumlah baris kumulatif dan II(-1; j) = 0 dan S(i; -1) = 0. Dengan menggunakan gambar integral, setiap jumlah persegi panjang dapat dihitung dalam empat referensi array seperti yang ditunjukkan pada Gambar 7.4, yang memberikan gagasan tentang bagaimana mengevaluasi jumlah piksel dari gambar asli di dalam persegi panjang D, yang dapat dihitung dengan empat referensi array. Nilai gambar integral di lokasi 1 adalah jumlah piksel dalam persegi panjang A. Nilai pada lokasi 2 adalah A + B, pada lokasi 3 adalah A + C dan pada lokasi 4 adalah A + B + C + D. Jumlah dalam D dapat dihitung sebagai 4 + 1 (2 + 3). Penjelasan lebih rinci diberikan pada bagian selanjutnya dan pada Gambar 7.5.



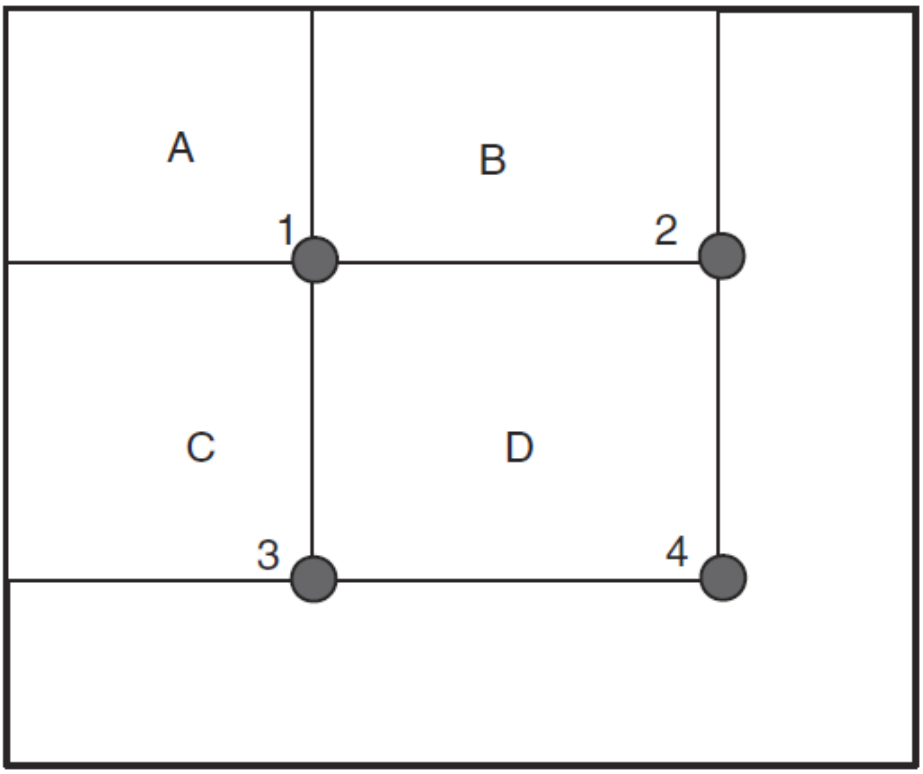
GAMBAR 7.5: Perhitungan fitur dua-persegi panjang dari array integral

Perhitungan fitur persegi panjang dari gambar integral

Perhitungan fitur dua-persegi panjang ditunjukkan pada Gambar 7.5, di mana fitur tworectangle ukuran 2x2 dipertimbangkan dan dihamparkan pada array integral,



GAMBAR 7.3: (a) Gambar asli, (b) gambar integralnya.



GAMBAR 7.4: Jumlah piksel dari gambar asli di dalam persegi panjang D dapat dihitung dengan empat referensi array

Menurut gambar integral dari bagian putih (a - c - d1 - b1) dari fitur persegi panjang, titik a = 5; b1 = 7; c = 8 dan d1 = 17. Untuk persegi panjang (a-c-d1-b1), jumlah dihitung sebagai d1+a-b1-c1 = 17+5-7-8 = 7 yang sama dengan 5 + 2 = 7 dalam array asli. Di dalam area hitam, jumlah persegi panjang juga dihitung. Bagian persegi panjang hitam (b1 - d1 - d2 - b2) dari fitur dua persegi panjang, d2 = 25; b2 = 10; b1 = 7; D1 = 17, dan jumlah persegi panjang dari area hitam ini dihitung sebagai D2+B1-B2-D1 = 25+7-10-17 = 5, yang sama dengan 4+1 = 5 pada gambar aslinya. Oleh karena itu, untuk fitur dua-persegi panjang, hanya enam referensi array (seperti 5; 7; 10; 8; 17; 25) yang diperlukan. Delapan referensi array diperoleh dalam kasus fitur tiga-persegi panjang dan sembilan untuk fitur empat-persegi panjang. Kumpulan fitur persegi panjang yang bermakna ini memiliki properti bahwa satu fitur dapat dievaluasi pada skala dan lokasi apa pun dalam beberapa operasi.

Detektor wajah yang efektif dapat dibangun dengan sedikitnya dua fitur persegi panjang. Mengingat keakuratan komputasi fitur-fitur ini, proses deteksi wajah dapat diselesaikan untuk seluruh gambar dengan kecepatan tinggi. Meskipun setiap fitur dapat dihitung dengan sangat efisien, menghitung set lengkap sangat mahal. Masalah ini ditangani dengan membentuk classifer yang efektif dengan menggabungkan fitur-fitur ini. Tantangan utamanya adalah menemukan fitur-fitur ini. Dalam metode deteksi wajah Viola-Jones, varian AdaBoost digunakan baik untuk memilih fitur dan untuk melatih classifer.

AdaBoost adalah algoritma pembelajaran dan digunakan untuk meningkatkan kinerja klasifikasi algoritma pembelajaran sederhana, dengan menggabungkan kumpulan fungsi klasifikasi yang lemah untuk membentuk pengklasifikasi yang lebih kuat. Dalam bahasa peningkatan, algoritma pembelajaran sederhana disebut pembelajar yang lemah karena kami tidak mengharapkan bahkan fungsi klasifikasi terbaik untuk mengklasifikasikan data pelatihan dengan baik. Untuk meningkatkan pelajar yang lemah, dipanggil untuk memecahkan serangkaian masalah belajar. Setelah putaran pertama pembelajaran, contoh-contoh tersebut ditimbang ulang untuk menekankan contoh-contoh yang salah diklasifikasikan oleh pengklasifikasi lemah sebelumnya. Pengklasifikasi kuat akhir mengambil bentuk kombinasi tertimbang dari pengklasifikasi lemah diikuti oleh ambang batas.

3.1.1.3 AdaBoost

Seperti yang telah dikatakan, AdaBoost adalah algoritma untuk membangun pengklasifikasi yang kuat sebagai kombinasi linier dari pengklasifikasi lemah sederhana. Klasifikasi akhir didasarkan pada suara tertimbang dari pengklasifikasi lemah. Pseudocode untuk AdaBoost diberikan sebagai berikut.

$Given\ (x_1, y_1), \ldots, (x_m, y_m)\ where\ x\ \in X\ and\ y\ \in Y = \{+1, -1\},$
$Initialize\ D_1(i) = {}^{1}/_{m}$
$For\ t = 1, \ldots . T$:

- *Train weak learner using distribution* D_t

- *Get weak hypothesis:* $h_t : X \rightarrow \{-1, +1\}$ *with error* $e_t : Pr_1\ D_t\ [h_t\ (x_i)\ \neq y_i]$
- $Choose\ a_t = {}^1/_{2ln}\ (\frac{1-e_t}{e_t})$
- $Update\ D_{t+1}(i) = \frac{D_t(i)}{Z_t} x e^{-a_t}\ if\ h_t(x_i) = y_i$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} x e^{a_t}\ if\ h_t(x_i) \neq y_i$$

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-a_t y_i h_t(x_i)\right)}{Z_t}$$

Di mana Z_t adalah faktor normalisasi (dipilih D_{t+1} sehingga menjadi distribusi). Output dari hipotesis akhir diberikan oleh

$$H(x) sign(\sum_{t=1}^{T} a_t h_t(x))$$

Algoritma mengambil sebagai input set pelatihan (x1; y1); :::; (xm; ym) di mana setiap xi milik beberapa domain atau ruang instance X dan setiap label yi berada di beberapa set label Y . AdaBoost memanggil algoritma pembelajaran lemah atau dasar yang diberikan berulang kali dalam serangkaian putaran t = 1,..., T. Salah satu ide utama dari algoritma ini adalah untuk mempertahankan distribusi atau set bobot di atas set pelatihan. Bobot distribusi ini pada contoh pelatihan i pada putaran t dilambangkan dengan Dt(i). Awalnya, semua bobot ditetapkan sama, tetapi pada setiap putaran, bobot contoh yang salah diklasifikasikan ditingkatkan sehingga pelajar yang lemah dipaksa untuk fokus pada contoh-contoh tinggi dalam set pelatihan. Tugas pembelajar lemah adalah mencari hipotesis lemah ht : X -> { +1} sesuai untuk distribusi Dt.

Algoritma AdaBoost yang dimodifikasi

Di antara semua fitur yang akan diakses, beberapa diharapkan untuk memberikan nilai tinggi yang hampir konsisten terutama ketika berada di atas wajah. Untuk menemukan fitur-fitur ini, Viola-Jones menggunakan versi modifikasi dari algoritma AdaBoost seperti yang diberikan di bawah ini.

Given the numbers of example images $(x_1, y_1), ..., (x_n, y_n)$ where $y_1 = 0, 1$ for negative and positive examples,
Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_1 = 0, 1$, where m and l are positive and negative examples.
For $t = 1, \cdots, T$:

- Normalize the weights:
$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
- Select the best weak classifier with respect to the weighted error:
$$\epsilon_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$$
- Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t and θ_t are the minimizers of ϵ_t.
- Update the weights:
$$w_{t+1,i} = w_{t,i} \beta^{1-e_t}$$
where $e_i = 0$ if example x_i is classified correctly and $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$C(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t; \\ 0 & otherwise \end{cases}$$

where $\alpha_t = log \frac{1}{\beta t}$

Bagian penting dari algoritma AdaBoost yang dimodifikasi berkaitan dengan penentuan fitur, polaritas, dan ambang batas terbaik. Untuk mencapai solusi cerdas untuk masalah ini, Viola-Jones menyarankan metode brute force sederhana. Ini berarti bahwa penentuan setiap pengklasifikasi lemah baru melibatkan evaluasi setiap fitur pada semua contoh pelatihan. Fitur berkinerja terbaik dipilih berdasarkan kesalahan tertimbang yang dihasilkannya. Dalam algoritma AdaBoost yang dimodifikasi, bobot contoh yang diklasifikasikan dengan benar berkurang sementara bobot contoh yang salah diklasifikasikan dijaga konstan. Akibatnya, lebih mahal untuk fitur kedua (dalam pengklasifikasi akhir) untuk salah mengklasifikasikan contoh. Dengan demikian fitur kedua dipaksa untuk fokus lebih keras pada contoh yang salah diklasifikasikan oleh yang pertama. Intinya adalah

bahwa bobot adalah bagian penting dari mekanisme algoritma AdaBoost. Selanjutnya, pengklasifikasi akhir adalah jumlah tertimbang dari pengklasifikasi lemah. Ini disebut lemah karena saja tidak dapat mengklasifikasikan gambar, tetapi bersama-sama dengan yang lain membentuk pengklasifikasi yang kuat.

PYTHON Code for AdaBoost classification

```
# AdaBoost classification example

from numpy import *

x= array([[0,1],[1,1],[2,1],[3,-1],[4,-1],\
[5,-1],[6,1],[7,1],[8,1],[9,-1]])
p = array([[0,0.1],[1,0.1],[2,0.1],[3,0.1],\
[4,0.1],[5,0.1],[6,0.1],[7,0.1],[8,0.1],[9,0.1]])
h_final =0
Thres = zeros((4,1))
alpha=zeros((4,1))

for t in range(0,3):
    err= zeros((9,1))
    thr = array([0.5,1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5]

    for k in range(0,9):
            if t ==2:
                h = sign(x[:,0]-thr[k])
            else:
                h = sign(thr[k]-x[:,0])

            for j in range(0,10):
                if h[j] != x[j,1]:
                    err[k] = err[k] + p[j,1]

    for l in range(0,9):
        if err[l] == err.min():
            indx = l
            break
    Thres[t]= thr[l]

    if t==2:
        h = sign(x[:,0]-thr[l])
    else:
        h = sign(thr[l]-x[:,0])

    alpha[t] = 0.5 * log((1-err.min())/err.min())
```

```
    q1 = exp(-alpha[t])
    q2 = exp(alpha[t])
    Zt = 2*sqrt(err.min()*(1-err.min()))


    for j in range(0,10):
        if h[j] == x[j,1]:
            p[j,1] = (q1*p[j,1])/Zt
        else:
            p[j,1] = (q2*p[j,1])/Zt


    f = alpha[t]*(h)
    h_final = h_final+f

decision = sign(h_final)
print decision
```

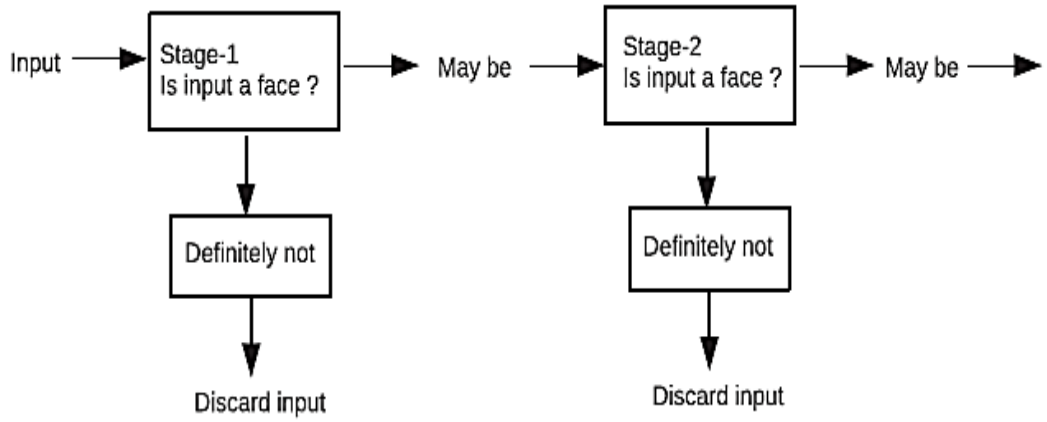
3.1.1.4 Cascade classifier

Bagian ini menjelaskan algoritma untuk membangun cascade pengklasifikasi yang mencapai peningkatan kinerja deteksi sementara secara proses mengurangi waktu komputasi. Pengklasifikasi yang lebih kecil, dan karenanya lebih efisien, dapat dibangun yang menolak banyak sub-jendela negatif sambil mendeteksi hampir semua contoh positif. Pengklasifikasi yang lebih sederhana digunakan untuk menolak sebagian besar sub-jendela sebelum pengklasifikasi yang lebih kompleks dipanggil untuk mencapai tingkat positif yang rendah.

Prinsip dasar dari algoritma deteksi wajah Viola-Jones adalah memindai detektor berkali-kali melalui gambar yang sama, setiap kali dengan ukuran baru. Bahkan jika gambar harus berisi satu atau lebih wajah, jelas bahwa sejumlah besar sub-jendela yang dievaluasi masih akan negatif (non-wajah). Kesadaran ini mengarah pada perumusan masalah yang berbeda. Alih-alih menemukan wajah, algoritma harus membuang non-wajah. Lebih cepat membuang non-wajah daripada menemukan wajah. Oleh karena itu, detektor yang hanya terdiri dari satu pengklasifikasi (kuat) tiba-tiba tampak tidak efisien karena waktu evaluasi konstan

tidak peduli bagaimana inputnya. Oleh karena itu kebutuhan akan pengklasifikasi berjenjang muncul.

Pengklasifikasi bertingkat terdiri dari tahapan, masing-masing berisi pengklasifikasi yang kuat. Tugas setiap tahap adalah untuk menentukan apakah sub-jendela yang diberikan jelas bukan wajah atau mungkin wajah. Ketika sub-jendela diklasifikasikan menjadi non-wajah pada tahap tertentu, sub-jendela itu segera dibuang. Sebaliknya, sub-jendela yang diklasifikasikan sebagai wajah mungkin diteruskan ke tahap berikutnya dalam kaskade. Oleh karena itu, semakin banyak tahapan yang dilewati sub-jendela tertentu, semakin tinggi kemungkinan sub-jendela berisi wajah. Konsep ini diilustrasikan dengan dua tahap pada Gambar 7.6. Dalam pengklasifikasi satu tahap, satu

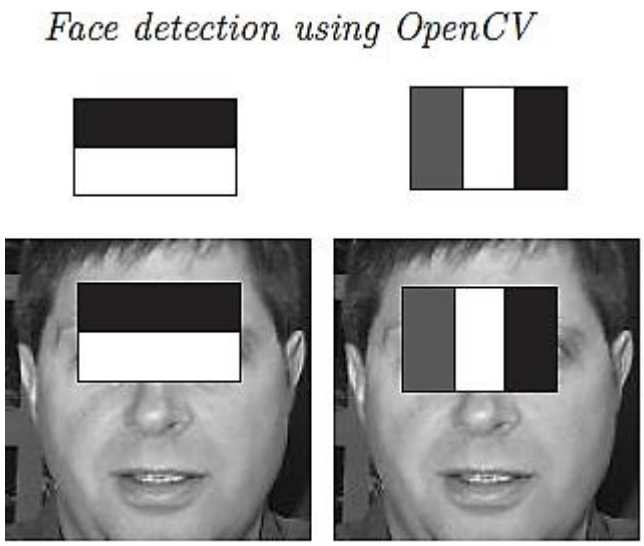


GAMBAR 7.6: Struktur pengklasifikasi kaskade

biasanya akan menerima buka negatif untuk mengurangi tingkat bukan positif. Namun, untuk tahap pertama dalam pengklasifikasi bertahap, bukan positif tidak dipertimbangkan karena tahap berikutnya diharapkan untuk menyelesaikannya. Oleh karena itu, banyak bukan positif pada tahap awal diterima. Akibatnya jumlah bukan negatif pada tahap akhir pengklasifikasi diharapkan menjadi sangat kecil. Oleh karena itu, lebih banyak perhatian (daya komputasi) diarahkan ke daerah gambar yang diduga mengandung wajah. Algoritma pelatihan untuk membangun detektor *cascade* dilakukan sebagai berikut:

- User selects values for f, the maximum acceptable false positive rate per layer, and d, the minimum acceptable detection rate per layer
- User selects target overall false positive rate, F_{target}
- P = set of positive examples
- N = set of negative examples
- $F_0 = 1.0; D_0 = 1.0$
- $i = 0$
- while $F_i > F_{target}$ $i \leftarrow i + 1$
  $n_i = 0; F_i = F_{i-1}$
  while $F_i > f \times F_{i-1}$

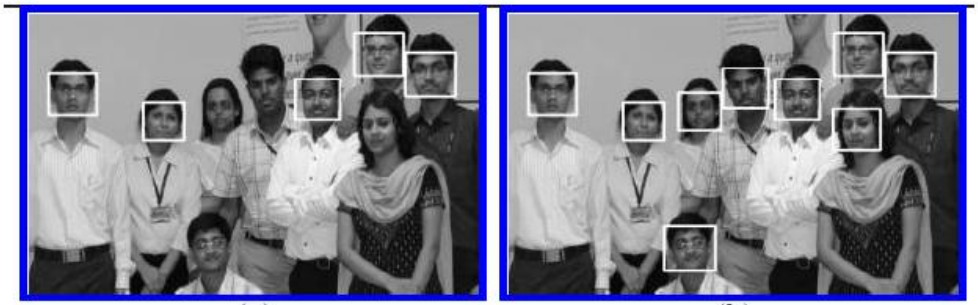
- $n_1 \leftarrow n_i + 1$
- Use P and N to train a classifier with n_i features using AdaBoost
- Evaluate current cascaded classifier on validation set to determine F_i and D_i



GAMBAR 7.7: Dua fitur yang diperoleh sebagai fitur terbaik dari AdaBoost

- Decrease threshold for the ith classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$ (this also affects F_i)
- $N \leftarrow \phi$
- If $F_i > F_{target}$ then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set N

Memperkenalkan pengklasifikasi kaskade alih-alih menerapkan semua fitur pada jendela, mungkin perlu mengelompokkan fitur ke dalam berbagai tahap pengklasifikasi dan menerapkannya satu per satu. Biasanya beberapa tahap pertama berisi jumlah fitur yang jauh lebih sedikit. Jika jendela gagal selama tahap pertama operasi, itu dibuang. Jika lolos, tahap kedua dilanjutkan. Jendela yang melewati semua tahap menunjukkan adanya daerah wajah. Dua fitur yang ditunjukkan pada Gambar 7.7 sebenarnya diperoleh sebagai dua fitur terbaik dari AdaBoost.



(a) (b)

FIGURE 7.8: Multiple face detections are shown in (a) and (b) by varying the threshold values.

The faces that are not detected in (a) are recognized as faces in (b) by simply changing the threshold.

PYTHON Code for OpenCV based face detection[1]

```
# OpenCV based face detection
import numpy as np
import cv2
face_cascade = cv2.CascadeClassifier \
("haarcascade_frontalface_alt2.xml")
img = cv2.imread("/home/pradipta/PRADIPTA/" \
"Database/CalTechfaces/16.jpg",0)
faces = face_cascade.detectMultiScale(img, 1.3,5)
for (x,y,w,h) in faces:
   cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
   roi_gray = img[y:y+h, x:x+w]
cv2.imshow("Faces",img)
```

3.1.2 Local Binary Pattern (LBP)

LBP memiliki keunggulan nyata seperti invarian rotasi dan invarian skala keabu-abuan, LBP adalah operator yang digunakan untuk menggambarkan fitur tekstur lokal gambar di bidang visi mesin. LBP merupaka operator tekstur yang sederhana namun sangat efektif, dan ketahanan perubahan skala abu-abu yang disebabkan oleh perubahan cahaya adalah atribut yang paling penting. Proses ekstraksi LBP adalah mengubah gambar asli menjadi diagram LBP, kemudian histogram LBP diagram LBP dihitung, dan gambar asli diwakili oleh histogram vektor ini. LBP dapat didefinisikan sebagai:

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} 2^p s(i_p - i_c)$$

Di mana (xc; yc) menunjukkan piksel pusat dengan intensitas ic dan ip adalah intensitas piksel tetangga. s adalah fungsi tanda yang didefinisikan sebagai:

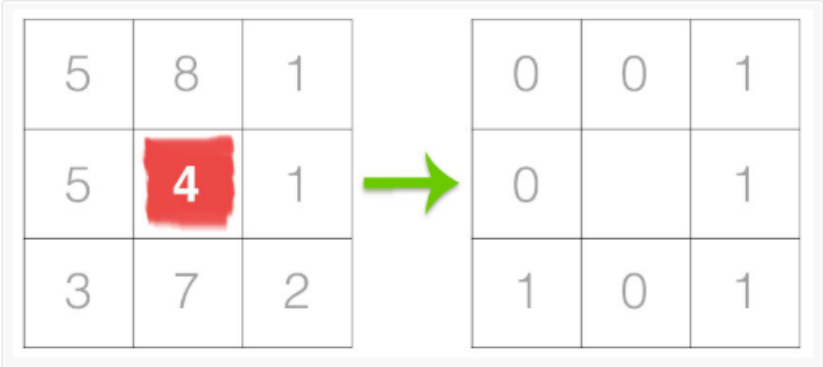
$$s(x) = \begin{cases} 1 & if \;\; x \geq 0 \\ 0 & else \end{cases}$$

Kemudian histogram gambar dihitung, setelah itu histogram dinormalisasi. Akhirnya, histogram statistik terhubung ke vektor eigen. Dengan menerapkan vektor eigen ke dalam proses SR, gambar HR yang direkonstruksi dapat mempertahankan detail tekstur dan informasi tepi dengan lebih baik.

Pada dasarnya algoritma ini akan mengesktrasi citra pada wajah kedalam fitur vektor dengan melakukan klasifikasi tekstur pada citra wajah manusia sehingga pengenalan objek menjadi lebih akurat(Purwati et al., 2018). Local Binary Pattern didefinisikan sebagai perbandingan nilai biner piksel pada pusat gambar dengan 8 nilai piksel disekelilingnya (Inen et al, 2011).
Langkah-langkah *Local Binary Pattern* yaitu:

1. Labeli piksel citra dengan melakukan proses thresholding ketetanggan 3x3 dari masing masing piksel sebagai nilai tengah dan mengubah hasilnya menjadi nilai biner. Setiap piksel dalam gambar *greyscale* pilih bagian berukuran *r* yang mengelilingi piksel tengah. Nilai LBP kemudian dihitung untuk piksel tengah dan disimpan dalam array dua dimensi dengan lebar dan tinggi yang sama dengan gambar input.

2. Jika intensitas piksel tengah lebih besar dari atau sama dengan tetangganya, maka tetapkan nilainya menjadi 1; jika tidak, atur ke 0 (Lihat Gambar 2).



Gambar 3.1.2. LBP dengan 8 piksel tetangga dan threshold

3. Untuk menghitung piksel tengah dapat dimulai dengan piksel tetangga manapun searah jarum jam maupun sebaliknya secara berurutan seperti pada Gambar 3 namun harus dilakukan secara konsisten untuk semua piksel.

Gambar 3.1.3 Menghitung piksel tengah