Sistem Deteksi Wajah Dengan Modifikasi Metode Viola Jones

Adinda Rizkita Syafira
Program Studi Informatika Universitas Muhammadiyah Surakarta (UMS) Surakarta, Indonesia
adinda.rizkita@gmail.com

Gunawan Ariyanto
Program Studi Informatika Universitas Muhammadiyah Surakarta (UMS) Surakarta, Indonesia
Gunawan.Ariyanto@ums.ac.id

Abstraksi— Deteksi wajah manusia merupakan salah satu topik yang paling banyak dipelajari di bidang computer vision. Tujuan deteksi wajah adalah untuk mengetahui ada atau tidaknya wajah pada suatu gambar. Meskipun tampak mudah dilakukan oleh manusia, ternyata pendeteksian wajah sangat rumit dilakukan oleh komputer karena terdapat beberapa kesulitan yang terkait dengan lokasi, sudut pandang, cahaya, dan oklusi. Penelitian ini menerapkan metode Viola Jones untuk membangun sistem deteksi wajah dengan bahasa pemrograman Python. Metode Viola Jones merupakan salah satu metode deteksi wajah dengan tingkat akurasi yang tinggi dan komputasi yang cepat. Metode Viola Jones menggunakan fitur Haar sebagai deskriptor kemudian menggabungkan Integral Image dan AdaBoost untuk mencari dan melakukan seleksi nilai fitur dan membentuk Cascade Classifier. Classifier tersebut yang akan digunakan untuk mendeteksi wajah pada gambar. Penelitian ini juga mengevaluasi tingkat akurasi sistem dengan cara memodifikasi nilai-nilai parameter yang ada di metode Viola Jones. Dari hasil pengujian menggunakan K-fold cross validation didapat hasil akurasi tertinggi sebesar 90,9% untuk gambar wajah dan 75,5% untuk gambar bukan wajah.Katakunci—deteksi wajah; python; viola jones

I. PENDAHULUAN

Salah satu topik penelitian bidang computer vision yang sering dipelajari beberapa dekade terakhir adalah deteksi wajah. Deteksi wajah digunakan untuk mengetahui ada atau tidaknya wajah pada suatu gambar dan bagian ini merupakan langkah pertama dalam proses identifikasi sehingga keberadaannya sangat vital. Tugas mendeteksi wajah sangat mudah bagi manusia; tetapi, tugas ini sangat rumit bagi komputer dikarenakan terdapat beberapa kompleksitas yang terkait dengan lokasi, sudut pandang, cahaya, dan oklusi [1]. Dari sekian banyak metode deteksi wajah, metode yang paling sering digunakan adalah metode Viola Jones. Metode Viola Jones diciptakan oleh Paul Viola dan Michael Jones pada tahun 2001. Proses deteksi wajah dilakukan dengan proses klasifikasi sebuah gambar berdasarkan nilai fitur sederhana melalui sebuah *classifier* yang dibentuk dari data *training* [2]. Metode ini menggabungkan konsep fitur Haar, *Integral Image*, dan *AdaBoost* yang kemudian diproses ke dalam bentuk *Cascade Classifier*. Bagian utama dari algoritma Viola Jones adalah komputasi dan seleksi fitur. Penggunaan *Integral Image* untuk mengekstrak fitur Haar telah mempercepat waktu komputasi dibandingkan dengan perhitungan per piksel. Keunggulan utama algoritma Viola Jones adalah sifatnya yang *robust*, yaitu mempunyai tingkat deteksi tinggi untuk pelacakan wajah dalam gambar dengan tingkat kesalahan yang rendah [3].

Tujuan penelitian ini untuk mengimplementasikan algoritme metode Viola Jones serta melakukan modifikasi yang diharapkan dapat meningkatkan performa sistem. Artikel ini menjelaskan prinsip kerja metode Viola Jones dan implementasi metode tersebut ke dalam sistem yang dibuat menggunakan bahasa pemrograman Python. Akan dijelaskan pula pengaruh dari

modifikasi nilai-nilai parameter yang ada di dalam metode Viola Jones sehingga di dapatkan sistem yang optimal dari sisi akurasi dan juga komputasi. Pengujian sistem ini menggunakan *K-fold cross validation* dengan cara mengamati tingkat akurasi untuk beberapa nilai parameter yang ada.

II. DASAR TEORI DAN TINJAUAN PUSTAKA

Deteksi wajah merupakan salah satu teknologi yang sering dimanfaatkan dan selalu dikembangkan seiring dengan perkembangan teknologi komputer. Saat ini banyak aplikasi komersial yang menggunakan algoritma pendeteksian wajah. Penelitian mengenai teknologi deteksi wajah maupun pengenalan wajah perlu dikembangkan lebih lanjut agar memperoleh hasil yang optimal. Kecepatan dan akurasi sistem pendeteksian wajah harus selalu ditingkatkan. Banyak dari sistem pendeteksian tersebut menggunakan metode Viola Jones sebagai metode pendeteksi objek [4].

Metode Viola Jones memiliki akurasi yang lebih tinggi dibandingkan metode deteksi wajah lainnya (semisal metode segmentasi, Euclidean distance, dan jaringan syaraf tiruan sederhana), namun metode Viola Jones juga memiliki kelemahan yaitu berupa kesulitan dalam menentukan wajah pada gambar wajah non-frontal (tidak tegak lurus ke arah kamera). Posisi wajah sangat menentukan keberhasilan metode Viola Jones dalam mendeteksi wajah [5].

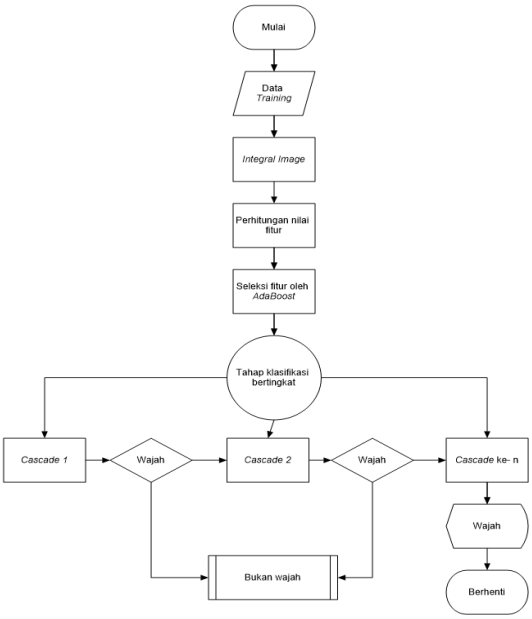
III. METODE PENELITIAN

Metode yang digunakan dalam penelitian ini adalah melakukan analisa prinsip kerja metode Viola Jones, kemudian mengumpulkan data set berupa gambar dan *source code*, melakukan modifikasi terhadap nilai variabel, melakukan pengujian sistem menggunakan *K-fold cross validation*, kemudian menganalisa kerja sistem tersebut.

A. Analisa Prinsip Kerja Metode Viola Jones

Alur proses sistem deteksi wajah dapat dilihat pada Gambar 1. Langkah pertama yang harus dilakukan dalam metode ini adalah mengubah gambar masukan menjadi representasi gambar baru berupa gambar integral. Gambar integral tersebut terdiri dari sejumlah rangkaian fitur yang cukup banyak. Fitur-fitur tersebut akan diseleksi oleh *AdaBoost* untuk dijadikan sebagai komponen classifier yang nantinya digunakan untuk mengklasifikasi gambar. Setelah itu *AdaBoost* akan melakukan training pada *classifier* yang telah dibentuk. Pada akhirnya, gambar akan diklasifikasi secara bertahap *(cascaded)* oleh *classifier* tersebut.

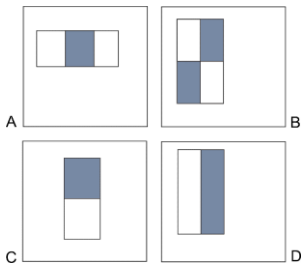
\



Gambar 1. Diagram alur proses deteksi wajah.

1. Fitur Haar

Fitur Haar merupakan fitur yang digunakan pada metode Viola Jones. Fitur ini terdiri dari satu nilai interval tinggi dan satu nilai interval rendah; untuk gambar dua dimensi disebut sebagai daerah terang dan daerah gelap. Fitur ini memiliki kelebihan berupa kinerja komputasinya yang sangat cepat.

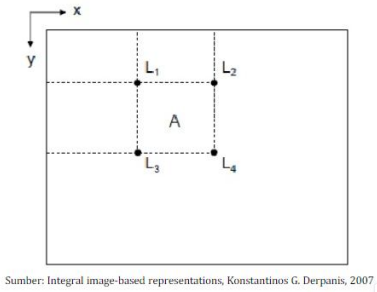


Gambar 2. Jenis fitur Haar.

Gambar 2 menunjukkan berbagai jenis fitur Haar dengan tiga jenis fitur berdasarkan jumlah persegi panjang yang terdapat didalamnya. Fitur pada bagian (C) dan (D) terdiri dari dua persegi panjang, bagian (B) terdiri dari 4 persegi panjang, dan bagian (A) terdiri dari tiga persegi panjang. Setiap fitur menghasilkan suatu nilai tunggal dan cara menghitung nilai fitur adalah dengan mengurangkan nilai piksel pada daerah terang dengan nilai piksel pada daerah gelap [6].

2. *Integral Image*

Integral Image merupakan suatu media yang digunakan untuk menghitung nilai fitur dengan cara mengubah gambar masukan menjadi suatu representasi gambar integral. Gambar integral akan menghasilkan suatu nilai fitur *Haar-like*. Gambar integral digunakan untuk menghitung jumlah semua piksel di dalam suatu persegi panjang dengan hanya menggunakan empat nilai secara efisien. Nilai-nilai tersebut adalah piksel pada gambar integral yang bertepatan dengan sudut-sudut persegi panjang pada gambar masukan [7] seperti yang terlihat pada Gambar 3.



Gambar 3. Representasi gambar integral

Untuk menghitung jumlah piksel A maka rumusnya adalah :

$$A = L_4 - (L_2 + L_3) + L_1 \quad (2)$$

Input Image

2	4	7	5	8
1	5	1	7	7
5	6	9	5	6
8	9	10	6	7
10	12	8	3	6

Integral Image

2	6	13	18	26
3	12	20	32	47
8	23	40	57	78
16	40	67	90	118
26	62	97	123	157

Gambar 4. Matriks *input image* menjadi *integral image*.

Seperti dilihat pada Gambar 4, setiap piksel dalam Integral Image harus sama dengan keseluruhan jumlah semua piksel di atas dan di sebelah kiri piksel yang bersangkutan.

Daerah terang

2	6	13	18	26
3	12	20	32	47
8	23	40	57	78
16	40	67	90	118
26	62	97	123	157

Daerah gelap

2	6	13	18	26
3	12	20	32	47
8	23	40	57	78
16	40	67	90	118
26	62	97	123	157

(a) (b)

Gambar 5. (a) Mencari jumlah fitur pada daerah terang. (b) Mencari jumlah fitur pada daerah gelap

Dari Gambar 5 dapat diketahui bahwa empat nilai yang digunakan untuk mencari nilai fitur adalah, L_1= 2, L_2 = 26, L_3 = 13, L_4 = 97 untuk daerah terang dan L_1 = 13, L_2 = 26, L_3 = 97, L_4 = 157 untuk daerah gelap. Menggunakan persamaan diatas maka, jumlah piksel pada daerah terang adalah 60 dan jumlah piksel pada daerah gelap adalah 47. Dengan mengurangi jumlah piksel pada daerah terang dan gelap maka didapat hasil 13 untuk nilai fitur pada persegi panjang tersebut.

3. *AdaBoost*

AdaBoost merupakan suatu algoritme *machine learning* yang bertujuan untuk melakukan seleksi secara spesifik terhadap fitur yang dianggap penting dan melakukan *training* pada beberapa *classifier* yang telah dibentuk. *AdaBoost* memiliki rangkaian filter yang cukup efisien untuk menggolongkan

daerah pada suatu gambar. Rangkaian filter tersebut terdiri dari *AdaBoost classifier* yang terbentuk dari gabungan *classifier lemah*. Suatu *classifier* dikatakan lemah jika, secara umum, tidak dapat memenuhi target klasifikasi yang telah ditentukan sebelumnya [8]. *Classifier* lemah tersebut menetapkan suatu bobot sehingga apabila digabungkan akan menjadi satu classifier yang kuat. Untuk memaksimalkan performa *AdaBoost* dalam sistem, Viola Jones menyarankan metode *brute force* yaitu, menentukan *classifier* lemah dengan cara mengevaluasi setiap fitur pada semua data *training* untuk menemukan fitur dengan kinerja terbaik. Namun hal ini diduga menjadi penyebab lamanya prosedur *training*.

4. Cascade Classifier

Cascade classifier merupakan metode klasifikasi yang bertugas untuk menghapus gambar bukan wajah dengan menggunakan *classifier* kuat yang telah di *training* oleh *AdaBoost* pada tiap tingkatan klasifikasi-nya.

Cascade Classifier merupakan metode klasifikasi bertingkat yang bertugas untuk menolak area gambar yang tidak terdeteksi wajah dengan menggunakan *classifier* yang telah dilatih oleh algoritme *AdaBoost* pada tiap tingkatan klasifikasinya. Pada klasifikasi tingkat pertama, tiap inputan berupa *sub-window* akan diklasifikasi secara sederhana. Seiring dengan bertambahnya tingkatan klasifikasi, maka diperlukan syarat yang lebih spesifik sehingga *classifier* / filter yang digunakan menjadi lebih kompleks. Hal ini ditujukan agar dapat mengurangi kasus wajah terdeteksi sebagai bukan wajah *(*kasus *false positive)*. Apabila terdapat input / *sub-window* yang gagal dilewatkan pada salah satu filter, maka area *sub-window* tersebut digolongkan sebagai bukan wajah. Namun apabila semua filter yang ada dalam rangkaian *cascade classifier* terlewati, maka area *sub-window* tersebut dianggap memiliki kemungkinan terdeteksi sebagai wajah [9]. Hasil dari klasifikasi ini berupa T (*True*) untuk gambar yang memenuhi nilai di semua *classifier* dan F (*False*) bila tidak memenuhi.

B. Pengumpulan data

Sistem yang dibangun menggunakan bahasa pemrograman Python dengan dataset berupa gambar wajah dan bukan wajah yang digunakan sebagai data *training* dan *testing*.

1. Data *Training*

Data *training* merupakan data yang digunakan untuk melakukan ekstraksi fitur yang kemudian dibentuk menjadi *classifier* untuk melakukan klasifikasi pada gambar. Data *training* merupakan dataset berisi *pre-processed* gambar *grayscale* wajah *(faces)* dan bukan wajah *(non-faces)* yang memiliki resolusi 19 x 19 piksel dengan format png. Dataset bersumber dari *The Center for Biological & Computational Learning* di *Massachusetts Institute of Technology* (CBCL MIT) yang diunduh dari situs https://github.com/INVASIS/Viola-Jones/tree/master/data.

2. Data *Testing*

Data *testing* merupakan data yang digunakan untuk melakukan pengujian pada sistem. Data *testing* secara otomatis dibuat melalui proses pengujian dengan metode *K-fold cross validation*.

3. *Source Code*

Source code merupakan implementasi algoritme deteksi wajah Viola Jones ke dalam bahasa Python. Kode program sebagian besar diadopsi dari Simon Hohberg, mahasiswa MIT melalui situs https://github.com/Simon-Hohberg/Viola-Jones/tree/master/violajones

C. Pengujian

Implementasi dan pengujian sistem dilakukan menggunakan *IDE JetBrains PyCharm Edu 3.0.1* dan bahasa pemrograman Python versi 3.6.2. Pengujian sistem menggunakan metode *K-fold cross*

validation. *K-fold* merupakan salah satu metode *Cross Validation* dengan melakukan partisi secara acak pada data *training* sebanyak nilai *K* dan melakukan percobaan sebanyak nilai tersebut pula. Masing-masing percobaan menggunakan data partisi ke-*K* sebagai data *testing* dan menggunakan sisa partisi lainnya sebagai data *training*. Metode ini diharapkan dapat meningkatkan akurasi hasil pengujian deteksi wajah.

Artikel ini juga menguji performa sistem berdasarkan nilai beberapa parameter dalam algoritme Viola Jones. Modifikasi dilakukan terhadap lima nilai parameter pada sistem yaitu,

a) num_classifier, adalah jumlah *classifier* kuat yang digunakan pada tahap *cascade*.
b) min_feature_height, adalah tinggi minimum fitur pada *sub-window*.
c) max_feature_height, adalah tinggi maksimal fitur pada *sub-window*.
d) min_feature_width, adalah lebar minimal fitur pada *sub-window*.
e) max_feature_width, adalah lebar maksimal fitur pada *sub-window*.

D. Analisa Kerja Sistem

Analisa kerja sistem dilakukan untuk mengukur performa berdasarkan pengujian sistem. Tingkat keberhasilan sistem dalam mendeteksi gambar wajah maupun bukan wajah akan ditampilkan dalam bentuk persen akurasi.

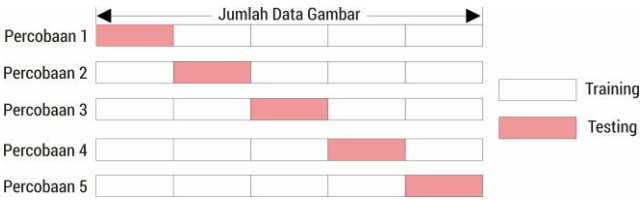
$$\frac{\text{jumlah gambar terdeteksi wajah}}{\text{jumlah data } \textit{testing} \text{ gambar wajah}} \times 100\ \% \quad \ldots\ldots\ldots\ldots\ldots\ldots(1)$$

$$\frac{\text{jumlah gambar terdeteksi bukan wajah}}{\text{jumlah data } \textit{testing} \text{ gambar bukan wajah}} \times 100\ \% \quad \ldots\ldots\ldots\ldots\ldots\ldots(2)$$

Persamaan 1 adalah rumus untuk mendapatkan persen akurasi tingkat wajah terdeteksi sebagai wajah (*true positive)* dan Persamaan 2 adalah rumus untuk mendapatkan persen akurasi tingkat bukan wajah terdeteksi sebagai bukan wajah (*true negative).* Hasil yang ditampilkan merupakan tingkat *true positive* dan tingkat *true negative* yang berhasil dideteksi oleh sistem.

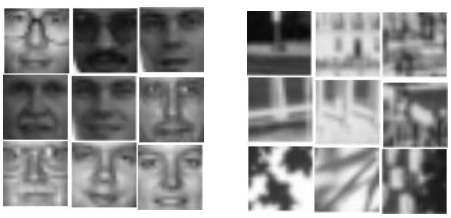
IV. HASIL DAN PEMBAHASAN

Dalam penelitian ini pengujian sistem menggunakan *K-fold cross validation* dengan dengan nilai *K* = 5. Setiap *fold* yang dijalankan bermakna satu kali proses *training* untuk seluruh dataset pada sistem maka pemilihan nilai *K* yang kecil (*K* = 5) bertujuan untuk mempersingkat waktu yang dibutuhkan untuk pengujian saat sistem dijalankan. Gambar 6 menjelaskan bahwa *K-fold cross validation* sangat memudahkan dalam melakukan partisi dataset gambar menjadi data *training* dan data *testing*.



Gambar 6. Proses *K-fold cross validation* dengan nilai *K* = 5.

Sistem menggunakan dataset gambar wajah dan bukan wajah seperti yang ditunjukkan pada Gambar 7 dengan jumlah yang sama untuk masing-masing gambar yaitu 2.425 kemudian dibagi menjadi data *training* sejumlah 1.940 gambar wajah dan bukan wajah dan data *testing* sebanyak 485 gambar wajah dan bukan wajah.



(a) (b)

Gambar 7. Data set gambar yang digunakan pada proses pengujian. (a) data set gambar wajah (b) data set gambar bukan wajah.

Kode program utama yang digunakan untuk implementasi sistem dapat dilihat pada

Gambar 8. Kode tersebut terdiri dari proses *training* sekaligus *testing*. Di dalamnya juga terdapat bagian untuk menghitung dan menampilkan hasil akurasi sistem.

```
import violajones.IntegralImage as ii
import violajones.AdaBoost as ab
import violajones.Utils as utils

if __name__ == "__main__":

    pos_training_path = 'trainingdata/faces'
    neg_training_path =
'trainingdata/nonfaces'

    #parameter yang dimodifikasi

    num_classifiers = 5
    min_feature_height = 8
    max_feature_height = 10
    min_feature_width = 8
    max_feature_width = 10

    #nilai K untuk pengujian menggunakan K-
fold cross        validation
    k = 5

    print('Loading faces..')
    x_data =
utils.load_images(pos_training_path)
    print('..done. ' + str(len(x_data)) + ' faces
loaded.\n\nLoading non faces..')
    y_data =
utils.load_images(neg_training_path)
    print('..done. ' + str(len(y_data)) + ' non
faces loaded.\n')

    num_val_samples_x = len(x_data) // k
    num_val_samples_y = len(y_data) // k
    print('Jumlah Data Faces :', len(x_data))
    print('Jumlah Data NonFaces :',
len(y_data))

    for i in range(k) :
        print('_____')
        print('processing fold #', i)

#mengubah input image menjadi integral
image
        x_val_data = x_data[i *
num_val_samples_x: (i + 1)*
num_val_samples_x]
        faces_ii_testing =
list(map(ii.to_integral_image, x_val_data))

        y_val_data = y_data[i *
num_val_samples_y: (i + 1) *
num_val_samples_y]
        non_faces_ii_testing =
list(map(ii.to_integral_image, y_val_data))

        x_train_data = x_data[:i *
num_val_samples_x]
        x_train_data.extend(x_data[(i + 1) *
num_val_samples_x:])
        faces_ii_training =
list(map(ii.to_integral_image, x_train_data))
y_train_data = y_data[:i *
num_val_samples_y]
        y_train_data.extend(y_data[(i + 1) *
num_val_samples_y:])
        non_faces_ii_training =
list(map(ii.to_integral_image, y_train_data))


        print('Val_Data_Faces:',
len(x_val_data))
        print('Val_Data_NonFaces:',
len(y_val_data))
        print('Training data faces: ',
len(x_train_data))
        print('Training nonfaces :',
len(y_train_data))


#melakukan tahap AdaBoost, membentuk
classifier
        classifiers = ab.learn(faces_ii_training,
non_faces_ii_training,
        num_classifiers, min_feature_height,
max_feature_height, min_feature_width,
max_feature_width)


#melakukan tahap cascade classifier
        print('Testing selected classifiers..')
        correct_faces = 0
        correct_non_faces = 0
        correct_faces =
sum(utils.ensemble_vote_all(faces_ii_testing
, classifiers))
        correct_non_faces = len(y_val_data) -
```

```
sum(utils.ensemble_vote_all(non_faces_ii_testing, classifiers))


#menampilkan hasil deteksi gambar wajah dan bukan wajah

print('..done.\n\nResult:\n      Faces: ' + str(correct_faces) + '/' + str(len(x_val_data))
          + '  (' + str((float(correct_faces) / len(x_val_data)) * 100) + '%)\n  non-Faces: '
          + str(correct_non_faces) + '/' + str(len(y_val_data)) + '  ('
          + str((float(correct_non_faces) / len(y_val_data)) * 100) + '%)')
```

Gambar 8. Kode program utama sistem deteksi wajah.

Jumlah *classifier* yang digunakan (nilai num_classifier) yaitu 3 dan 5 pada masing-masing percobaan sistem. Nilai num_classifier dibatasi maksimum 5 dikarenakan waktu untuk menjalankan sistem menjadi sangat lama apabila lebih dari nilai tersebut.

Selain menggunakan *K-fold cross validation*, dilakukan juga pengujian secara manual dengan membagi dataset ke dalam dua kelompok yaitu data *training* dan data *testing* secara manual. Dalam hal ini dipilih secara manual dan acak 80 % dataset menjadi data *training* dan sisanya 20 % menjadi data *testing*. Tabel 1 dan 2 menggambarkan ringkasan hasil pengujian.

Tabel 1. Hasil dari pengujian sistem tanpa *K-fold cross validation*

Classifier	Minimum feature width/height	Maximum feature width/height	Faces (%)	Non-faces (%)
3	6	8	68,6	65,5
	8	10	81,4	54,4
	10	12	87	57,1
5	6	8	88,5	52,2
	8	10	86,8	47,2
	10	12	87	57,1

Tabel 2. Hasil dari pengujian sistem menggunakan *K-fold cross validation*

Classifier	Minimum feature width/height	Maximum feature width/height	Faces (%)	Non-faces (%)
3	6	8	85,2	65,6
	8	10	82,6	75,5
	10	12	72,5	69,4
5	6	8	89,4	70,7
	8	10	90,9	68,4
	10	12	84,9	62

Pada Tabel 1 menampilkan hasil percobaan tanpa menggunakan *K-fold cross validation*. Dapat kita lihat bahwa percobaan yang menggunakan tiga *classifier* menunjukkan tingkat *true positive* pada gambar wajah mencapai 68-87% dan tingkat *true negative* pada gambar bukan wajah mencapai 54-65% .Sedangkan percobaan yang menggunakan lima *classifier* menunjukkan tingkat *true positive* pada gambar wajah mencapai 86-88% dan tingkat *true negative* pada gambar bukan wajah mencapai 47-57%.

Hasil percobaan menggunakan *K-fold cross validation* dapat dilihat pada Tabel 2. Percobaan dengan *K-fold cross validation* mengambil nilai rata-rata dari hasil akurasi setiap *fold* sebagai nilai akhir yang ditampilkan. Hasil percobaan menunjukkan bahwa dengan tiga *classifier* menghasilkan tingkat *true positive* pada gambar wajah mencapai 72-85,2 % dan tingkat *true negative* pada gambar bukan wajah mencapai 65-75,5 %. Sedangkan percobaan yang menggunakan lima *classifier* menunjukkan tingkat *true positive* pada gambar wajah mencapai 84-90,9 % dan tingkat *true negative* pada gambar bukan wajah mencapai 62-70,7 %.

Berdasarkan percobaan pada Tabel 1 dan Tabel 2 maka dapat diambil kesimpulan bahwa hasil tertinggi didapat dari pengujian menggunakan *K-fold cross validation* untuk pendeteksian gambar wajah adalah 90,9% menggunakan parameter berupa lima *classifier*, nilai minimum tinggi/lebar fitur sebesar 8 piksel, dan nilai maksimum tinggi/lebar fitur sebesar 10 piksel. Hasil tertinggi untuk pendeteksian gambar bukan wajah adalah 75,5% menggunakan parameter berupa 3 *classifier*, nilai minimum tinggi/lebar fitur sebesar 8 piksel, dan nilai maksimum tinggi/lebar fitur sebesar 10 piksel.

Semakin banyak jumlah *classifier* yang digunakan, maka semakin tinggi tingkat akurasi sistem. Banyaknya jumlah *classifier* juga berpengaruh terhadap jumlah tingkatan pada tahap *cascade classifier*. Semakin banyak jumlah *classifier* maka sistem menjadi semakin teliti dalam melakukan deteksi wajah dikarenakan *classifier* berusaha untuk mengurangi tingkat *false positive* pada *sub-window*, sehingga hal tersebut menyebabkan kemungkinan tingkat deteksi wajah meningkat [10].

V. KESIMPULAN

Penelitian ini berhasil mengimplementasikan metode Viola Jones ke dalam sistem deteksi wajah dengan menggunakan bahasa pemrograman Python. Sistem menunjukkan bahwa *classifier* yang terbentuk dari proses *training* berhasil melakukan tugas deteksi wajah dengan baik.

Pengujian sistem deteksi wajah dengan menggunakan *K-fold cross validation* menghasilkan tingkat akurasi yang mencapai 90,9 % untuk gambar wajah dan 75,5 % untuk gambar bukan wajah. Dari hasil pengujian tersebut didapat kesimpulan bahwa modifikasi nilai parameter terbaik untuk melakukan deteksi gambar wajah adalah num_classifier=5, min_feature width/height=8, dan max_feature_width/height=10, sedangkan nilai parameter terbaik untuk melakukan deteksi gambar bukan wajah adalah num_classifier=3, min_feature_width/height=8, dan max_feature_width/height=10.

DAFTAR PUSTAKA

[1] K. Cen, "Study of Viola-Jones Real Time Face Detector."

[2] I. S. Nugraha and Muljono, "Aplikasi Android Deteksi Mata Menggunakan Metode Viola-Jones," *Univ. Dian Nuswantoro, Semarang*, 2015.

[3] H. Pratikno, "Kontrol Gerakan Objek 3D Augmented Reality Berbasis Titik Fitur Wajah dengan POSIT," *Jnteti*, vol. 4, no. 1, pp. 16–24, 2015.

[4] D. A. Prasetya and I. Nurviyanto, "Deteksi wajah metode viola jones pada opencv menggunakan pemrograman python," *Simp. Nas. RAPI XI FT UMS*, pp. 18–23, 2012.

[5] M. D. Putro, T. B. Adji, and B. Winduratna, "Sistem Deteksi Wajah dengan Menggunakan Metode Viola-Jones," *Sci. Eng. Technol.*, pp. 1–5, 2012.

[6] Y.-Q. Wang, "An Analysis of the Viola-Jones Face Detection Algorithm," *Image Process. Line*, vol. 4, pp. 128–148, 2014.

[7] O. H. Jensen, "Implementing the Viola-Jones Face Detection Algorithm," Technical University of Denmark, 2008.

[8] R. E. Schapire, "A Brief Introduction to Boosting Generalization error," *Ijcai 99*, pp. 1401–1406, 1999.

[9] P. Viola and M. Jones, "Robust Real-time Object Detection," vol. 57, no. 2, pp. 1–25, 2001.

[10] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proc. 2001 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition. CVPR 2001*, vol. 1, p. I-511-I-518, 2001.