Jurnal Informatika, Vol. 12, No. 2, November 2014, 96-108 DOI: 10.9744/informatika.12.2.96-108
ISSN 1411-0105

SISTEM DETEKSI WAJAH PADA *OPEN SOURCE PHYSICAL COMPUTING*

Yupit Sudianto[1*], Febriliyan Samopa[1]
Jurusan Sistem Informasi Fakultas Teknologi Informasi ITS
Kampus ITS, Sukolilo, Surabaya, 60111
Telp: (031) 5922949, Fax: (031) 5964965
E-mail: ic.yupit@gmail.com, samopa@gmail.com
*Korespondensi penulis

Abstrak: Dalam beberapa tahun terakhir, deteksi wajah merupakan salah satu area penelitian yang menarik. Berbagai algoritma telah diteliti, namun mayoritas penelitian tersebut masih diimplementasikan pada komputer. Pembangunan sistem deteksi wajah pada komputer memerlukan biaya investasi yang tidak sedikit, selain harus mengeluarkan biaya pengadaan komputer, juga diperlukan biaya untuk operasional seperti penggunaan listrik, dimana komputer membutuhkan daya/watt yang besar, dengan daya yang besar tentunya biaya operasional yang harus dikeluarkan juga besar. Untuk menghemat kedua biaya tersebut dalam penelitian ini diusulkan pembangunan sistem deteksi wajah dengan menggunakan arduino. Sistem deteksi wajah yang diusulkan pada penelitian ini bersifat autonomous (mandiri), dengan kata lain peran komputer akan digantikan oleh arduino. Tipe Arduino yang digunakan adalah arduino Mega 2560 dengan spesifikasi mikrokontroler ATMEGA 2560, kecepatan 16 MHz, flash memory 256 KB, SRAM 8 KB, dan EEPROM 4 KB. Dengan spesifikasi tersebut, tidak semua algoritma deteksi wajah dapat diimplementasikan pada arduino. Untuk mengatasi keterbatasan memori yang dimiliki oleh arduino akan digunakan metode template matching dengan menggunakan fitur wajah berupa template yang berbentuk seperti topeng. Berdasarkan hasil uji coba yang telah dilakuan dalam penelitian ini detection rate yang berhasil dicapai adalah sebesar 80%-100%. Dimana, keberhasilan arduino dalam mengidentifikasi wajah dipengaruhi oleh jarak antara wajah manusia dengan kamera dan pergerakan manusia..

Kata kunci: *Arduino,* deteksi wajah*, embedded system.*

Abstract: *Face detection is one of the interesting research area. Majority of this research implemented on a computer. Development of face detection on a computer requires a significant investment costs. In addition to having to spend the cost of procurement of computers, is also required for operational cost such as electricity use, because the computer requires large power/watt.This research is proposed to build a face detection system using Arduino. The system will be autonomous, in other word the role of computer will be replaced by Arduino. Arduino is used is Arduino Mega 2560 with specifications microcontroller AT MEGA 2560, a speed of 16 MHz, 256 KB flash memory, 8 KB SRAM, 4 KB EEPROM. So not all face detection algorithm can be implemented on the Arduino. The limitations of memory owned by the arduino will be resolved by applying the method of template matching using the facial features in the form of a template that is shaped like a mask. Detection rate achieved in this study is 80% - 100%. Where, in the Arduino's success in identifying the face are influenced by the distance between the camera with the human face and human movement.*

Keywords*: Arduino, face detection, embedded system.*

PENDAHULUAN

Deteksi wajah merupakan proses untuk mengidentifikasi wajah pada sebuah gambar, yang dipengaruhi oleh beragam latar belakang dan pencahayaan, berbagai ekspresi wajah, gaya rambut, aksesoris seperti kumis, jenggot, kacamata maupun aksesoris yang lain [18]. Berbagai bidang yang memanfaatkan deteksi wajah adalah pengenalan wajah atau *face recognition*, kamera pengawasan, pengenalan ekspresi, fotografi, dan lain sebagainya. Pada bidang pengenalan wajah dan kamera pengawasan diperlukan sistem yang dapat berjalan 24 jam 7 kali seminggu.

Sistem deteksi wajah yang diimplementasikan pada komputer memerlukan investasi yang besar. Selain diperlukan biaya untuk pengadaan komputer, juga diperlukan biaya operasional (seperti penggunaan listrik, karena komputer membutuhkan daya/watt yang besar). Sistem deteksi wajah yang diimplementasikan pada komputer dengan prosesor Inter Core 2 Quad membutuhkan daya sebesar 95

Watt, sedangkan jika prosesor Pentium 4 yang digunakan daya yang dibutuhkan sebesar 84 Watt [7]. Untuk mengatasi permasalahan tersebut, maka pada penelitian ini diusulkan pembangunan sistem deteksi wajah dengan menggunakan *open source physical computing*. *Open source physical computing* yang digunakan dalam penelitian ini adalah arduino Mega 2560.

Fokus pada penelitian ini adalah mencari algoritma deteksi wajah yang dapat diimplementasikan pada arduino Mega 2560. Prosesor yang digunakan arduino Mega 2560 adalah ATmega 2560 dengan kecepatan sebesar 16 MHz. Sedangkan memori yang dimiliki arduino Mega 2650 adalah *flash memory* 256 KB (dengan 8 KB yang digunakan oleh *bootloader*), 8 KB dari SRAM, dan 4 KB EEPROM. Karena keterbatasan tersebut, maka tidak semua algoritma deteksi wajah dapat diiimplementasikan pada arduino Mega 2560.

PENGENALAN DETEKSI WAJAH

Hingga saat ini berbagai teknik untuk mengidentifikasi wajah telah diusulkan, teknik tersebut terbagi menjadi empat kategori yaitu metode *knowledge-based*, metode *feature-invariant*, metode *template-matching*, dan metode *appearance-based* [36]. Kelemahan dan kelebihan dari masing-masing metode dapat dilihat pada Tabel 1.

Dalam sistem deteksi wajah digunakan dua rumus untuk melakukan validasi dari sistem yang dibangun, yaitu *detection rate* dan *false positive*. *Detection rate* merupakan kemampuan keberhasilan dalam mendeteksi wajah, yang dihitung dengan menggunakan rumus 1

$$\text{detection rate} = \frac{\text{jumlah wajah yang berhasil dideteksi}}{\text{jumlah wajah yang diujikan}} \text{x } 100\% \quad (1)$$

Sedangkan *false positive* merupakan kegagalan menolak bukan wajah, yang dihitung dengan menggunakan rumus 2.

$$\text{false positive} = \frac{\text{jumlah bukan wajah yang dianggap wajah}}{\text{jumlah total window yang diuji}} \text{x } 100\% \quad (2)$$

Untuk mengetahui kinerja dari sistem deteksi wajah dilakukan perhitungan kecepatan deteksi (fps), yang dihitung dengan rumus 3.

$$\text{kecepatan deteksi} = \frac{\text{frames}}{\text{waktu saat ini - waktu mulai}} \text{x } 100\% \quad (3)$$

DETEKSI WAJAH PADA EMBEDDED SYSTEM

Pada penelitian ini, pembangunan sistem deteksi wajah pada arduino akan mengacu pada berbagai penelitian tentang sistem deteksi wajah pada *embedded system* yang berhasil mencapai kecepatan yang tinggi. Diantaranya, berhasil mencapai kecepatan sebesar 30 fps dengan ukuran gambar

Tabel 1. Kekurangan dan Kelebihan dari metode deteksi wajah

Metode	Kelebihan	Kekurangan
Knowledge Based	• Proses identifikasi wajah dilakukan dengan menggunakan fitur yang sederhana yaitu keberadaan mata yang simetris, hidung, dan mulut. • Detection rate yang berhasil dicapai adalah 86,5.	• Semakin ketat aturan yang digunakan, kemungkinan terjadinya kegagalan dalam mengidentifikasi wajah juga semakin besar. Sebaliknya, jika aturan yang digunakan terlalu umum, maka false positive yang dihasilkan akan semakin besar. • Posisi wajah harus dalam posisi frontal. • Metode ini berjalan dengan optimal jika pada citra hanya mengandung satu wajah (*face localization*).
Feature Invariant	• Dapat mengidentifikasi wajah dalam berbagai posisi kemiringan dan berbagai kondisi cahaya. • Detection rate yang dicapai adalah 85% hingga 100%.	• Semakin ketat aturan yang digunakan, kemungkinan terjadinya kegagalan dalam mengidentifikasi wajah juga semakin besar. Sebaliknya, jika aturan yang digunakan terlalu umum, maka false positive yang dihasilkan akan semakin besar. • Metode ini berjalan dengan optimal jika pada citra hanya mengandung satu wajah (*face localization*).
Template Matching	• Mudah diiimplementasikan. • Detection rate yang berhasil dicapai 70% hingga 87.5%.	• Sulit untuk mendapatkan bentuk *template* wajah yang umum. • Posisi wajah harus dalam posisi frontal.
Appearance Based	• Fitur yang digunakan banyak sehingga hasilnya lebih akurat. • Dapat mengidentifikasi wajah dengan berbagai posisi kemiringan. • Detection rate yang berhasil dicapai 90% hingga 100%.	• Memerlukan waktu yang lama untuk melakukan proses training. • Memerlukan banyak alokasi memori karena fitur yang digunakan banyak.

sebesar 320x240 piksel [8]. Penelitian ini menggunakan metode yang berdasarkan pada warna kulit dan *lines-of-face template*. Penelitian lain dengan menggunakan algoritma ANN, kecepatan yang berhasil dicapai pada penelitian ini sebesar 625 fps dengan ukuran gambar sebesar 640x480 piksel [7]. Selain itu, dengan menggunakan algoritma AdaBoost, kecepatan yang berhasil dicapai adalah 64-139 fps dengan gambar sebesar 1024x768 piksel [11]. Yutong mengusulkan metode yang berdasarkan warna kulit, dan kecepatan yang berhasil dicapai adalah sebesar 60 fps dengan ukuran gambar sebesar 1280x1024 piksel [38]. Dengan menggunakan *edge information,* dan kecepatan yang berhasil dicapai adalah 28-60 fps untuk gambar dengan ukuran 320x240 piksel [11]. Dari berbagai penelitian tersebut akan dipilih algoritma mana yang paling sesuai dengan arduino. Penelitian yang dilakukan oleh Yang mengenai karakteristik metode-metode deteksi wajah, akan dijadikan sebagai dasar untuk memilih algoritma yang akan diimplementasikan pada arduino [36].

ARDUINO

Ada banyak platform mikrokontroler selain arduino seperti i-cubeX, Arieh Robotics Project Junior, Dwengo, EmbeddedLab, GP3 yang menawarkan fungsionalitas yang sejenis. Namun Arduino memiliki beberapa keunggulan diantara:

- Perangkat keras bersifat *open source*. Diagram rangkaian elektronik dari arduino disediakan secara gratis untuk semua orang, sehingga dapat dirangkai sendiri oleh siapa saja tanpa harus membayar kepada pembuat arduino.
- Software bersifat *open source*. Perangkat lunak arduino diterbitkan sebagai *open source* dan ketersediaan *extension* di berikan oleh programmer yang telah berpengalaman.
- *Cross Platform*. Software arduino dapat berjalan di sistem operasi Windows, Macinctosh OSC, dan Linux. Beberapa sistem mikrokontroler yang lain terbatas di windows.
- Lebih murah jika dibandingkan dengan platform mikrokontroler yang lain, karenahardwarenya yang bersifat *open source* sehingga setiap pengguna dapat merangkai arduino sendiri.

Arduino adalah *platform* dari *physical computing* yang bersifat *open source* [1]. Pengertian *physical computing* adalah sistem atau perangkat fisik yang dibangun dengan menggunakan perangkat keras dan perangkat lunak, yang bersifat interaktif, yaitu dapat menerima rangsangan dari lingkungan dan memberikan respon balik. Konsep *physical computing* diterapkan dalam desain alat yang menggunakan sensor dan mikrokontroler untuk menerjemahkan input analog ke dalam sistem perangkat lunak, yang kemudian digunakan untuk mengontrol gerakan alat-alat seperti lampu, motor, dan lain sebagainya. Tipe arduino yang digunakan dalam penelitian ini adalah arduino Mega 2560.

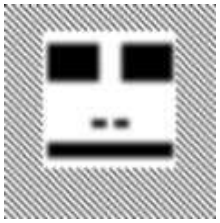
Spesifikasi dari arduino Mega 2560 adalah prosesor ATmega 2560 dengan kecepatan sebesar 16 MHz, flash memory 256 KB, SRAM 8 KB, EEPROM 4 KB, jumlah pin digital IO/PWM 54/15, dan jumlah pin analog 16.

SERIAL KAMERA OV7670 WITHOUT FIFO

Serial kamera yang digunakan dalam penelitian ini adalah serial kamera versi *ov7670 without FIFO*. Berikut ini adalah beberapa alasan penggunaan serial kamera *ov7670 without FIFO*:

- Tegangan yang dibutuhkan rendah yaitu sebesar 3,3 Volt, sehingga cocok untuk digunakan pada aplikasi portable.
- Harga relatif terjangkau.
- Tetap memiliki kepekaan yang baik, meskipun digunakan di ruangan yang cahayanya kurang.
- Standard antarmuka SCCB kompatibel dengan antarmuka I2C interface.
- Mendukung format data Raw RGB, RGB (GRB 4:2:2, RGB565/555/444), YUV (4:2:2) dan YCbCr (4:2:2).
- Ukuran data gambar yang didukung adalah VGA (640x480), QVGA (320x240), CIF (352x240), QCIF (176x144), dan manual penskalaan yang dapat diperkecil hingga ukuran 40x30.

Untuk petunjuk penggunaan serial kamera ini dapat dilihat pada "*OV7670/0V7171 CMOS VGA (640X480) CameraChip™ Implementation Guide*" [21].



Gambar 1. Bentuk template wajah yang digunakan sebagai fitur

IMPLEMENTASI ALGORITMA DETEKSI WAJAH PADA KOMPUTER

Algoritma deteksi wajah yang diusulkan dalam penelitian ini mengacu pada metode *template*

matching, dimana identifikasi wajah dilakukan dengan menggunakan fitur berupa *template* wajah yang dapat dilihat pada Gambar 1. Pada wajah manusia bagian mata, hidung, dan mulut berwarna lebih gelap dari pada bagian yang lain seperti pipi maupun dahi. Sehingga bagian mata, hidung, dan mulut dari fitur yang digunakan pada penelitian ini diberi nilai 0, sedangkan bagian yang lain bernilai 255.

Secara garis besar proses identifikasi wajah yang diusulkan dalam penelitian ini dilakukan dengan cara menghitung perbedaan nilai antara fitur dengan citra. Pada dasarnya jika perbedaan nilai antara citra dan fitur lebih kecil dari nilai *threshold* maka dalam citra tersebut merupakan wajah. Sedangkan jika perbedaan nilai tersebut lebih besar dari nilai *threshold* maka citra tersebut bukan wajah manusia.

Pada umumnya kondisi obyek yang ditangkap oleh kamera dipengaruhi oleh berbagai kondisi pencahayaan yang berbeda. Kondisi cahaya yang terlalu terang mengakibatkan citra berwarna lebih terang, sedangkan kondisi cahaya yang terlalu gelap mengakibatkan citra berwarna lebih gelap. Serta posisi datangnya cahaya juga mempengaruhi kondisi warna dari citra sehingga tidak menutup kemungkinan ada bagian citra yang berwarna terang pada salah satu sisi dan berwarna gelap pada sisi yang lain. Untuk mengatasi permasalahan tersebut maka diusulkan untuk menggunakan *histogram equalization*.

Secara runut konsep identifikasi wajah yang diusulkan dalam penelitian ini dijelaskan pada Gambar 2.

Gambar 2. Konsep identifikasi wajah yang diusulkan

Proses identifikasi wajah dilakukan dengan menggunakan citra *grayscale*, kemudian dilakukan *histogram equalization* terhadap citra tersebut. Langkah selanjutnya menghitung perbedaan nilai citra *grayscale* yang telah melalui proses *histogram equalization* dengan fitur wajah. Langkah berikutnya adalah melakukan identifikasi wajah, identifikasi wajah dilakukan dengan cara membandingkan antara nilai yang didapat dari perbedaan antara fitur dengan citra, dibandingkan dengan nilai *threshold*, jika nilai tersebut lebih kecil dari *threshold* maka pada citra tersebut terdapat wajah, namun jika nilai tersebut lebih besar dari *threshold* maka pada citra tersebut tidak terdapat wajah. Implementasi konsep tersebut meliputi dua tahap yaitu tahap training dan tahap deteksi.

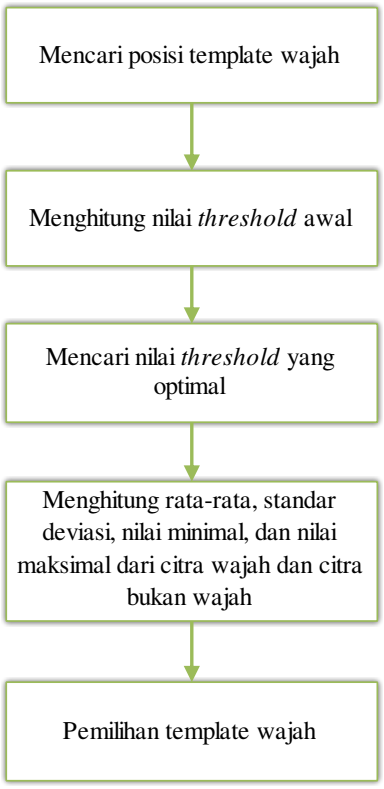
Training

Tahap *training* atau pelatihan dilakukan terlebih dahulu sebelum tahap deteksi wajah. Tujuan dari tahap training adalah untuk mendapatkan *template* wajah yang dapat menghasilkan *detection rate* paling tinggi dan *false positive* paling rendah. Tahap *training* dilakukan dengan menggunakan 2000 citra wajah dan 4000 citra bukan wajah yang berukuran 19x19 piksel. Contoh citra wajah yang digunakan dapat dilihat pada gambar 3, sedangkan contoh citra bukan wajah dapat dilihat pada gambar 4. Secara umum langkah-langkah yang dilakukan pada tahap *training* dapat dilihat pada gambar 5 yang dijelaskan sebagai berikut:

a. Mencari posisi *template*
   Untuk mendapatkan posisi baris dan kolom yang optimal, setiap template wajah di-training dengan 2000 citra wajah. Tahap ini dilakukan per *template* wajah. *Pseudocode* untuk mendapatkan posisi baris dan kolom dapat dilihat pada tabel 2. Contoh template wajah yang digunakan dapat dilihat pada gambar 6.
b. Menghitung nilai *threshold* awal
   Setelah mendapatkan koordinat baris dan kolom dari *template* wajah, langkah selanjutnya adalah menghitung nilai *threshold* awal. Nilai *threshold* tersebut akan digunakan pada tahap berikutnya yaitu mencari nilai *threshold* yang optimal. Proses pencarian nilai *threshold* awal dilakukan per *template* wajah dengan menggunakan seluruh citra wajah dan citra bukan wajah. Algoritma yang digunakan untuk menghitung nilai *threshold* awal dapat dilihat pada Tabel 3.

Gambar 3 Citra wajah

Gambar 4. Citra bukan wajah



Gambar 5. Langkah-langkah untuk melakukan training

Gambar 6. Template wajah

Tabel 2 *Pseudocode* untuk mencari posisi *template*

- Inisialisasi jumlah baris dan kolom dari citra wajah serta fitur wajah.
  (a,b) ← size(W)
  (c,d) ← size (P)
  Dimana, W adalah fitur wajah, a dan b adalah jumlah baris dan kolom dari fitur wajah. Sedangkan, P adalah citra wajah, c dan d adalah jumlah baris dan kolom untuk citra wajah.
- Inisialisasi untuk menyimpan total nilai fitur hasil per koordinat (y,x) dari citra wajah.
  FOR y ← 0 TO c-a
  FOR x ← 0 TO d-b
  valueWindow(y,x) ← 0
  END-FOR
  END-FOR
- Menghitung nilai fitur per fitur wajah dengan menggunakan citra wajah.
  FOR i ← TO j
  S ← histeq (Pi)
  FOR y ← 0 TO c
  FOR x ← 0 TO d
  Z ← W – S(y:y+a-1 , x:x+b-1)
  v ← sum(sum(Z))
  valueWindow(y,x) ← valueWindow(y,x)+v
  END-FOR
  END-FOR
  END-FOR
  (q,r) ← min(min(valueWindow))
  Dimana, histeq adalah *histogram equalitzation*, histeq yang digunakan pada penelitian ini dijelaskan pada bab 2.4. Sedangkan j adalah jumlah citra wajah yang digunakan untuk *training*. Untuk q adalah koordinat baris dari posisi fitur. Dan r adalah koordinat kolom dari posisi fitur.

Tabel 3. *Pseudocode* untuk menghitung nilai *threshold* awal

- Inisialisasi untuk menyimpan nilai fitur citra wajah..
  FOR y ← 0 TO j
  valuePos(y) ← 0
  END-FOR
  Dimana, j adalah jumlah citra wajah yang digunakan dalam *training*.
- Inisialisasi untuk menyimpan nilai fitur citra bukan wajah.
  FOR y ← 0 TO l
  valueNeg(y) ← 0
  END-FOR
  Dimana, l adalah jumlah citra bukan wajah yang digunakan untuk *training*.
- Menghitung nilai fitur dengan cara mengurangkan fitur wajah dengan data *training*, baik citra wajah maupun citra bukan wajah. Bagian data *training* yang dikurangkan dengan fitur wajah disesuaikan dengan koordinat fitur wajah yang telah didapatkan dari algoritma pada tahap 1.

```
(a,b) ← size(W)
FOR y ← 0 TO j
    S ← histeq(P_y)
    Z ← W – S(q:q+a-1 , r:r+b-1)
    v ← sum(sum(Z))
    valuePos(y) ← v
END-FOR
FOR y ← 0 TO l
    S ← histeq(N_y)
    Z ← W – S(q:q+a-1 , r:r+b-1)
    v ← sum(sum(Z))
    valueNeg(y) ← v
END-FOR
threshold ← (mean(valuePos)+mean(valueNeg))/2
```

Dimana, W adalah fitur wajah, sedangkan a dan b adalah baris dan kolom dari fitur wajah. Untuk q dan r adalah koordinat baris dan kolom dari fitur wajah pada window. P adalah citra wajah, j adalah jumlah citra wajah. N adalah citra bukan wajah, dan l adalah jumlah citra bukan wajah.

c. Mencari nilai *threshold* yang optimal
Nilai *threshold* yang telah didapatkan sebelumnya adalah *threshold* awal. Pada bagian ini nilai *threshold* tersebut akan diolah sehingga didapatkan *threshold* yang optimal. Nilai *threshold* yang optimal adalah nilai *threshold* yang memiliki *true negative* dan *false positive* yang paling kecil. Proses pencarian nilai *threshold* ini dilakukan per *template* wajah, untuk lebih jelasnya dapat dilihat pada tabel 4.

Tabel 4. *Pseudocode* untuk mencari nilai *threshold* yang optimal

- Inisialisasi untuk mengolah *false positive* dan *true negative*.

```
FP ← 0
TN ← 0
checkErr ← 0
```

- Mencari nilai *threshold* yang optimal dengan cara menghitung *false positive* dan *true negative* dari citra wajah dan citra bukan wajah.

```
WHILE TRUE
    FOR y ← 0 TO j
        IF valuePos(y) > threshold
            TN ← TN+1
        END-IF
    END-FOR
    FOR y ← 0 TO l
        IF valueNeg(y) <= threshold
         FP ← FP+1
        END-IF
    END-FOR
    temp ← (TN+FP)/(j+l)
    TN ← TN/j
    FP ← FP/l
    IF TN > FP
        threshold ← threshold +1
    ELSE IF TN < FP
        threshold ← threshold-1
    END-IF
    IF sumErr > temp
        sumErr ← temp
        checkErr ← 0
    END IF
    IF checkErr == 20
        break;
    ELSE
        checkErr ← checkErr+1
    END-IF
    END-WHILE
```

Dimana, TN adalah *true negative* dan FP adalah *false positive*. Sedangkan j dan l adalah jumlah citra wajah dan citra bukan wajah. Untuk checkErr digunakan untuk melihat perkembangan sumErr. Nilai sumErr didapatkan dari (FP+TN)/(j+l). Jika nilai sumErr tidak menurun hingga 20 iterasi berikutnya, maka nilai *threshold* yang optimal telah ditemukan.

d. Analisa citra wajah dan bukan wajah
Bagian ini digunakan untuk menghitung nilai minimal, maksimal, rata-rata, dan standard deviasi dari data wajah dan bukan wajah. Perhitungan nilai minimal, maksimal, rata-rata, dan standar deviasi dilakukan dengan menggunakan fungsi yang disediakan di Matlab. Nilai-nilai tersebut dan nilai *threshold* yang optimal yang telah didapatkan pada tahap sebelumnya digunakan untuk mendapatkan nilai *threshold* yang dapat digunakan untuk mendeteksi wajah.

e. Pemilihan template wajah
Pemilihan *template* wajah merupakan langkah terakhir dalam tahap training. Pada langkah ini akan dipilih *template* wajah yang akan digunakan untuk melakukan deteksi wajah. Pemilihan *template* wajah dilakukan dengan berdasar pada *true negative* dan *false positive* yang diperoleh dari hasil *training*. Perhitungan *true negative* dilakukan dengan menggunakan rumus 1, sedangkan *false positive* dihitung dengan menggunakan rumus 2. Semakin kecil nilai *true negative* dan *false positive* maka semakin baik. Jika dari hasil *training* diperoleh lebih dari satu *template* yang memiliki *true negative* dan *false positive* yang kecil, maka akan dilakukan uji coba terhadap *template* tersebut. Tujuan dari uji coba tersebut adalah untuk melihat *detection rate* yang dihasilkan oleh setiap *template*. *Template* yang memiliki *detection rate* paling tinggi akan digunakan sebagai fitur untuk melakukan deteksi wajah. Jika ternyata template tersebut memiliki *detection rate* yang sama baiknya, maka akan dipilih berdasarkan *false positive* dari hasil uji coba yang paling rendah.

$$\text{false positive} = \frac{\text{total false positive}}{\text{jumlah citra bukan wajah}} \text{ x } 100\% \quad (4)$$

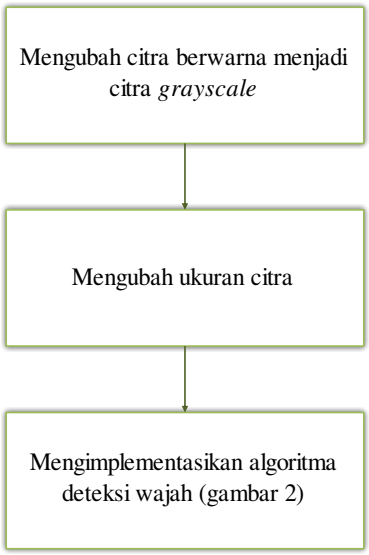
$$\text{true negative} = \frac{\text{total true negatve}}{\text{jumlah citra wajah}} \quad (5)$$

Deteksi Wajah

Setelah tahap *training* selesai dilakukan dan diperoleh *template* wajah yang memiliki *detection rate* paling tinggi dan *false positive* paling rendah, maka akan dilakukan tahap deteksi wajah. Secara garis besar langkah-langkah yang dilakukan pada tahap deteksi wajah yang ditunjukkan pada Gambar 7 adalah sebagai berikut:

a. Mengubah citra berwarna menjadi citra grayscale
   Mengubah citra berwarna menjadi citra grayscale dengan rumus 3.6.
   $Y = 0.299\ Red - 0.587\ Green - 0.114\ Blue$ (6)
b. Mengubah ukuran citra
   Mengubah ukuran citra menjadi sebesar 40x30 piksel dengan menggunakan fungsi "imresize" yang telah disediakan oleh matlab.
c. Mengimplementasikan algoritma deteksi wajah.
   Proses identifikasi wajah dilakukan dengan cara mengambil potongan citra atau yang biasa disebut sebagai *window* dengan ukuran 19x19 piksel. Kemudian dilakukan *histogram equalization* pada *window* tersebut. Selanjutnya menghitung perbedaan nilai antara *window* hasil *histrogram equalization* dengan fitur wajah. Jika nilai perbedaan tersebut kurang dari *threshold* maka citra pada *window* tersebut terdapat wajah, sedangkan jika nilai perbedaan tersebut lebih besar dari *threshold* maka tidak terdapat wajah pada *window* tersebut. Proses ini berlangsung dari posisi piksel yang paling atas, kemudian bergerak dari kiri ke kanan, kemudian dilanjutkan ke posisi satu piksel dibawahnya, lalu bergerak lagi dari kiri ke kanan, dan begitu seterusnya hingga seluruh citra dengan ukuran 40x30 piksel diproses seluruhnya. Jika ditemukan lebih dari satu lokasi yang teridentifikasi sebagai wajah maka dipilih lokasi yang memiliki nilai yang paling kecil.

Hasil implementasi dari langkah-langkah pada Gambar 7 dapat dilihat pada gambar 8. Implementasi tersebut diawali dengan membaca citra berwarna (Gambar 8 bagian a), kemudian mengubahnya menjadi citra *grayscale* (Gambar 8 bagian b), kemudian mengubah ukuran citra *grayscale* menjadi citra *grayscale* yang berukuran 40x30 piksel (Gambar 8 bagian c), selanjutnya melakukan identifikasi wajah terhadap citra tersebut yang dimulai dari koordinat (1,1) kemudian bergerak ke kanan hingga koordinat (1,22) kemudian bergerak satu baris dibawahnya hingga total 264 window teridentifikasi (Gambar 8 bagian d), window yang teridentifikasi sebagai wajah akan disimpan dalam memori dan tidak menutup kemungkinan bahwa ada lebih dari satu window yang teridentifikasi sebagai wajah (Gambar 8 bagian e), sehingga langkah terakhir dicari nilai identifikasi wajah yang paling kecil (Gambar 8 bagian f).

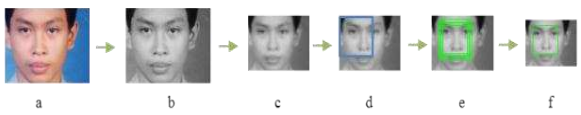


Gambar 7. Langkah-langkah untuk melakukan identifikasi wajah

Tabel 5. *Pseudocode* untuk mengidentifikasi wajah

```
FOR y ← 0 TO N
    FOR x ← 0 TO M
    window ← IMG(y:y+(c-1) , x:x+(d-1))
    window ← histeq(window)
    Z ← W – window(q:q+(a-1) , r:r+(b-1))
    tempValue ← sum(sum(Z))
    IF tempValue < threshold
    IF value > tempValue
    value ← tempValue
    END-IF
    END-IF
    END-FOR
END-FOR
```

Dimana, q dan r adalah koordinat baris dan kolom dari fitur wajah. Sedangkan a dan b adalah jumlah baris dan kolom dari fitur wajah.



Gambar 8. Hasil implementasi algoritma deteksi wajah pada komputer

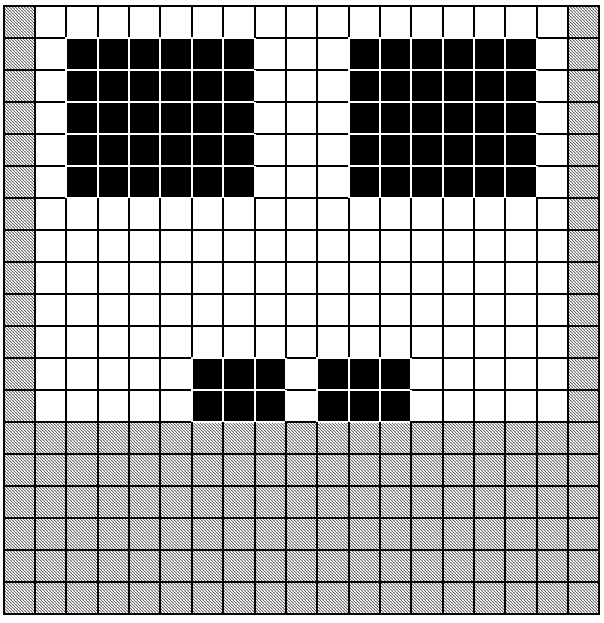
Uji Kebenaran dan Uji Kinerja

Uji kebenaran dan uji kinerja yang dijelaskan pada bagian ini adalah uji kebenaran dan uji kinerja yang dilakukan pada saat algoritma deteksi wajah diimplementasikan dengan menggunakan komputer. Pengujian tersebut dilakukan dengan menggunakan serangkaian skenario uji coba. Skenario tersebut disusun untuk mengetahui kemampuan dari algoritma deteksi wajah dalam mengidentifikasi keberadaan wajah yang dipengaruhi oleh pencahayaan, warna dari latar belakang, dan posisi wajah baik miring maupun frontal. Kemampuan algoritma deteksi wajah dalam mengidentifikasi wajah dilihat dari nilai *detection rate*, *false positive*, dan kecepatan yang dihasilkan pada saat uji coba. Berikut ini adalah skenario uji coba yang dilakukan:

a. Skenario pemilihan *threshold*
   Skenario ini dilakukan untuk mendapatkan rentang nilai *threshold* yang memiliki *detection rate* paling tinggi dan *false positive* paling rendah. Rentang nilai *threshold* tersebut diperoleh dengan menguji coba berbagai rentang nilai dengan berdasarkan pada nilai *threshold* hasil *training* serta nilai minimal, maksimal, rata-rata, dan standar deviasi dari data wajah dan bukan wajah. Selain untuk mendapatkan rentang nilai *threshold*, skenario ini juga bertujuan untuk mengetahui kemampuan dari algoritma deteksi wajah yang telah dipilih terhadap citra dengan posisi wajah frontal dan citra bukan wajah. Uji coba ini dilakukan dengan menggunakan 100 citra berupa foto orang indonesia dengan posisi wajah frontal dan 100 citra berupa foto bukan wajah. Citra wajah dan bukan wajah yang digunakan tersebut memiliki berbagai kondisi pencahayaan yang berbeda dan warna latar belakang yang berbeda.
   Berdasarkan uji coba yang telah dilakukan, template wajah yang terdiri dari mata dan hidung adalah template wajah yang memiliki kemampuan identifikasi yang paling baik. Bentuk template wajah tersebut dapat dilihat pada Gambar 9. Sedangkan posisi template tersebut pada *window* dapat dilihat pada Gambar 10. *Template* tersebut memiliki *threshold* sebesar 23006, sedangkan untuk nilai minimal, maksimal, rata-rata, dan standar deviasi untuk data wajah dan bukan wajah dapat dilihat pada Tabel 6. Untuk hasil dari skenario pemilihan *threshold* dapat dilihat pada Tabel 7.
b. Skenario posisi wajah miring
   Tujuan dari skenario ini adalah untuk mengetahui apakah nilai *threshold* yang telah dipilih dari skenario pemilihan *threshold* tetap memiliki kemampuan identifikasi wajah yang baik pada saat wajah dalam keadaan miring. Skenario ini dilakukan dengan menggunakan 50 foto orang indonesia dengan posisi wajah miring. Kemiringan posisi wajah didapatkan dengan menggunakan aplikasi IrfanView. Posisi kemiringan yang diuji coba pada skenario ini adalah 1°, 2°, 5°, dan 10°.
   Pada bagian ini dilakukan uji coba untuk mengidentifikasi wajah dengan posisi miring. Uji coba ini dilakukan dengan menggunakan 50 foto orang indonesia. Hasil uji coba dari skenario posisi wajah miring dapat dilihat pada Tabel 8. Sedangkan citra hasil uji coba dapat dilihat pada Gambar 11.

Uji kinerja dilakukan dengan cara menghitung kecepatan dari algoritma yang diusulkan untuk melakukan identifikasi wajah terhadap data wajah dan data bukan wajah. Data yang digunakan adalah citra berwarna dengan ukuran 320x240 piksel. Berdasarkan hasil uji coba, semakin banyak bagian yang teridentifikasi sebagai wajah, maka kecepatan yang dihasilkan juga akan semakin lambat. Selama proses uji coba, kecepatan algoritma untuk melakukan identifikasi wajah adalah sebesar 7-18 fps.

Gambar 9. Bentuk template wajah yang digunakan



Gambar 10. Posisi template wajah pada gambar 11 pada window

Tabel 6. Nilai *template* wajah

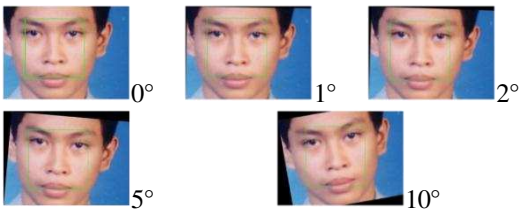
	Wajah	Bukan wajah
Minimal	14173	17654
Maksimal	30715	36375
Rata-rata	19892	26360
Standar deviasi	2299.8	2787.2

Tabel 7. Skenario pemilihan *threshold*

No	*Threshold*	*Detection Rate*	*False Positive*
1	*Threshold* > 14173 & *Threshold* < 17654	55%	0.38% - 2.65%
2	***Threshold* > 14173 & *Threshold* < 19892**	**100%**	**0.38% - 9.47%**
3	*Threshold* > 14173 & *Threshold* < 23006	100%	0.38% - 23.48%
4	*Threshold* > 17593 & *Threshold* < 23006	100%	0,38% - 23,11%

Tabel 8. Hasil uji coba dengan berbagai kemiringan

No	Skenario	*Detection Rate*
1	Posisi kemiringan wajah sebesar 1°	98%
2	Posisi kemiringan wajah sebesar 2°	98%
3	Posisi kemiringan wajah sebesar 5°	98%
4	Posisi kemiringan wajah sebesar 10°	84%



Gambar 11. Hasil identifikasi wajah dengan berbagai posisi wajah.

IMPLEMENTASI ALGORITMA DETEKSI WAJAH PADA ARDUINO

Sistem deteksi wajah pada penelitian ini melibatkan tiga perangkat keras yaitu serial kamera ov7670 *without FIFO*, arduino Mega 2560, dan servo S3003. Rangkaian dari tiga perangkat keras tersebut dapat dilihat pada Gambar 12.

Urutan proses deteksi wajah dapat dilihat pada Gambar 13. Secara garis besar urutan proses tersebut dijelaskan sebagai berikut:

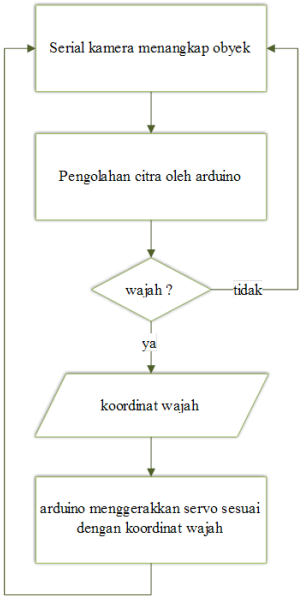
1. Pengambilan obyek
   Proses pengambilan obyek dilakukan oleh serial kamera ov7670 *without FIFO*. Agar serial kamera dapat berjalan, serial kamera tersebut dihubungkan dengan arduino. Untuk koneksi antara serial kamera dengan arduino dapat dilihat pada Tabel 9. Khusus untuk pin XCLK, pin tersebut dihubungkan dengan system clock. Desain system clock pada penelitian ini dapat dilihat pada Gambar 14. Untuk proses pengambilan obyek dari serial kamera ov7670 *without FIFO* dijelaskan pada Gambar 15 dan Gambar 16. Hasil dari proses pengambilan obyek adalah citra, dimana citra tersebut nantinya akan diolah dengan menggunakan algoritma deteksi wajah. Berdasarkan Gambar 15 dan Gambar 16, untuk mendapatkan citra perlu diperhatikan nilai VSYNC, HREF, dan PCLK. Baik nilai VSYNC, HREF maupun PCLK selalu berubah antara low dan high. Proses pengambilan obyek dimulai pada saat VSYNC bernilai *low* (Gambar 16). Setelah VSYNC bernilai *low* langkah selanjutnya adalah melihat kondisi HREF dan PCLK. Setiap citra terdiri dari baris dan kolom. Proses untuk memperoleh nilai dari setiap piksel dimulai pada saat HREF bernilai *high*. Banyaknya jumlah HREF yang bernilai *high* selama VSYNC bernilai *low* adalah sesuai dengan banyaknya baris dari citra. Sedangkan PCLK dilihat pada saat HREF sedang bernilai *high*. Nilai dari setiap piksel didapatkan pada saat PCLK bernilai *high* (Gambar 15). Banyaknya jumlah PCLK yang bernilai *high* adalah sebanyak jumlah kolom dari citra. Sehingga secara garis besar nilai setiap piksel dari citra didapatkan pada saat kondisi VSYNC bernilai *low*, HREF bernilai *high*, dan PCLK bernilai *high*. Jika nilai piksel tidak didapatkan pada saat kondisi tersebut maka nilai yang didapatkan adalah nilai yang salah. Citra yang diambil adalah citra *grayscale* (*luminance*) dengan ukuran 40x30 piksel.

Gambar 12. Sistem deteksi wajah yang dibangun dengan arduino mega 2560, serial kamera ov7670 without FIFO, dan servo

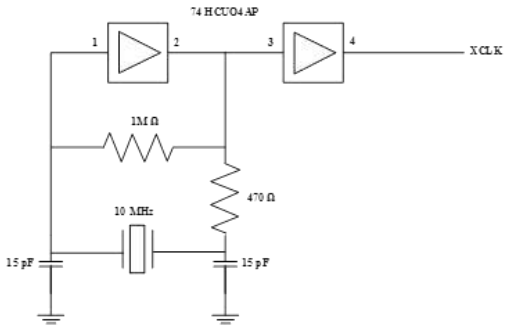
2. Pengolahan citra oleh arduino
   Setelah mendapatkan citra dari serial kamera, citra tersebut akan diolah oleh algoritma deteksi wajah. Algoritma deteksi wajah yang digunakan

dapat dilihat pada Tabel 5. Algoritma tersebut akan mengidentifikasi keberadaan dan posisi wajah dari citra. Jika pada citra tersebut terdapat wajah, maka arduino akan menggerakkan servo secara horisontal dan vertikal untuk mengejar posisi wajah, proses ini akan berlangsung terus-menerus selama pada citra tersebut terdapat wajah. Namun, jika tidak terdapat wajah pada citra tersebut, maka servo akan kembali ke posisi normal.

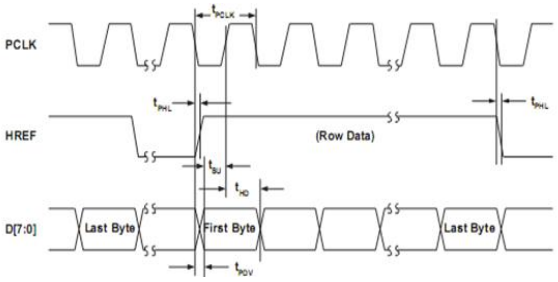
Hasil implementasi dari langkah-langkah yang terdapat pada Gambar 13 dapat dilihat pada Gambar 17. Sistem deteksi wajah pada arduino diawali dari pengambilan obyek oleh serial kamera ov7670 *without FIFO* (Gambar 17 bagian a), dimana hasil dari pengambilan obyek dapat dilihat pada Gambar 17 bagian b, kemudian dilakukan identifikasi wajah yang dimulai dari koordinat (0,0) (koordinat array pada arduino dimulai dari (0,0)) kemudian identifikasi wajah bergerak ke kanan hingga koordinat (0,21) lalu bergerak satu piksel ke bawah hingga total 264 window teridentifikasi (Gambar 17 bagian c), window yang teridentifikasi sebagai wajah akan disimpan dalam SRAM dan tidak menutup kemungkinan bahwa ada lebih dari satu window yang teridentifikasi sebagai wajah (Gambar 17 bagian d), kemudian dicari nilai identifikasi wajah yang paling kecil (Gambar 17 bagian e), selanjutnya dengan menggunakan koordinat dari nilai identifikasi wajah yang paling kecil arduino menggerakan servo menyesuaikan koordinat tersebut sehingga untuk pengambilan obyek berikutnya posisi wajah berada di tengah kamera (Gambar 17 bagian f).



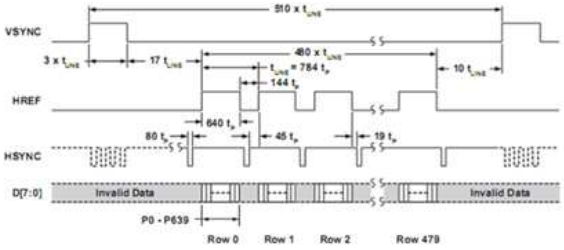
Gambar 13. Urutan proses deteksi wajah dengan menggunakan arduino



Gambar 14. Desain system clock



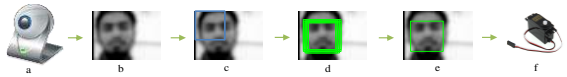
Gambar 15. Horizontal timing (Omnivision, 2006)



Gambar 16. Frame timing (Omnivision, 2006)

Tabel 9. Koneksi pin serial kamera ov7670 *without FIFO* dengan arduino

No.	PIN Serial Kamera	TIPE	KET	PIN ARDUINO
1	VDD	*Supply*	*Power supply*	3,3V
2	GND	*Supply*	*Ground*	*Ground*
3	SIOC	Input	SCCB clock	SIOC, dengan *pull up* 4k7Ω atau 10k Ω
4	SIOD	Input/output	SCCB data	SIOD, dengan *pull up* 4k7Ω atau 10k Ω
5	VSYNC	Output	*Vertical synchronization*	Digital pin
6	HREF	Output	*Horizontal synchronization*	Digital pin
7	PCLK	Output	*Pixel clock*	Digital pin
8	XCLK	Input	*System clock*	-
9	D0-D7	Output	*Video parallel output*	Digital pin
10	RESET	Input	*Power supply*	3,3V
11	PWDN	Input	*Ground*	*Ground*



Gambar 17. Hasil implementasi algoritma deteksi wajah pada arduino

UJI COBA DAN EVALUASI

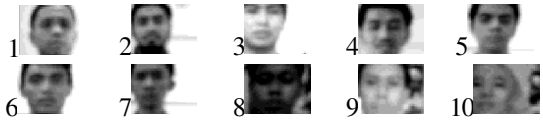
Uji coba pada bagian ini meliputi uji kebenaran dan uji kinerja. Pengujian tersebut dilakukan dengan menggunakan serangkaian skenario. Skenario tersebut disusun untuk mengetahui kemampuan dari algoritma deteksi wajah pada saat diimplementasikan dengan menggunakan arduino. Kemampuan dari algoritma deteksi wajah dilihat dari nilai *detection rate*, *false positive*, dan kecepatan yang dihasilkan pada saat uji coba. Implementasi algoritma deteksi wajah pada arduino melibatkan tiga perangkat keras yaitu serial kamera, arduino, dan servo. Sehingga, skenario uji coba akan difokuskan pada kemampuan tiga perangkat keras tersebut. Berikut ini adalah skenario uji coba yang dilakukan:

a. Skenario posisi wajah frontal dan miring
   Tujuan dari skenario ini adalah untuk mengetahui apakah algoritma deteksi wajah yang diimplementasikan pada arduino dapat mengidentifikasi wajah dengan posisi frontal dan miring. Pada bagian ini juga dilakukan pengujian jarak antara wajah dengan kamera. Uji coba pada bagian ini dilakukan dengan menggunakan 10 orang indonesia dengan 3 posisi wajah frontal dan 2 posisi wajah miring. Wajah yang digunakan sebagai uji coba dapat dilihat pada Gambar 18.
b. Skenario mendeteksi koordinat wajah
   Skenario ini dilakukan untuk menguji apakah arduino dapat menggerakkan servo sehingga posisi wajah yang telah teridentifikasi sebelumnya, berada di daerah tengah dari serial kamera. Pada bagian ini uji coba dilakukan dengan menggunakan 10 orang indonesia dengan 5 kali pengambilan obyek. Setiap kali pengambilan obyek akan dilihat apakah arduino dapat menggerakkan servo sesuai dengan koordinat yang didapat.
c. Skenario wajah yang bergerak
   Skenario ini dilakukan untuk menguji kemampuan arduino dalam mendeteksi wajah yang bergerak. Uji coba pada bagian ini menggunakan 10 orang indonesia. Dimana, uji coba dilakukan sebanyak 5 kali pengambilan obyek dengan posisi manusia yang berpindah-pindah.

Hasil dari skenario tersebut dapat dilihat pada Tabel 10. Berdasarkan uji coba tersebut, sistem deteksi wajah yang diusulkan dalam penelitian ini cukup baik untuk mengidentifikasi wajah manusia dan mampu mengikuti keberadaan wajah, baik wajah tersebut dalam kondisi diam maupun bergerak, dengan syarat pergerakan wajah tersebut tidak cepat. Posisi wajah yang mampu diidentifikasi adalah posisi frontal dan posisi miring, untuk posisi miring, kemiringan maksimal adalah 10°, jika lebih dari 10° posisi wajah tersebut akan susah teridentifikasi. Hasil uji coba dengan skenario posisi wajah frontal dan miring dapat dilihat pada Gambar 19, sedangkan hasil uji coba untuk skenario mendeteksi koordinat wajah dapat dilihat pada Gambar 20, dan untuk hasil uji coba skenario wajah yang bergerak dapat dilihat pada Gambar 21. Keberhasilan dalam identifikasi wajah juga dipengaruhi oleh jarak antara wajah dengan kamera, berdasarkan Gambar 18, jarak yang diperlukan oleh setiap wajah dengan kamera adalah sebagai berikut:

- wajah nomor 1 dan 2 memerlukan jarak 85 cm,
- wajah nomor 3 memerlukan jarak 80 cm
- wajah nomor 4, 5 , 6, dan 7 memerlukan jarak 70 cm
- wajah nomor 8, 9, dan 10 memerlukan jarak 65 cm

Sedangkan jarak vertikal antara kamera dengan ujung kepala adalah 30 cm. Luas daerah yang dijangkau oleh kamera adalah sebesar 25x19 cm.

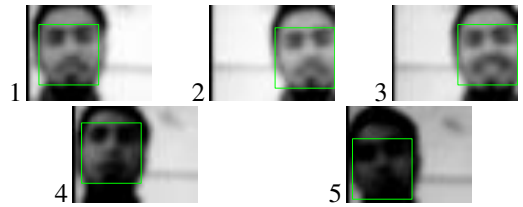


Gambar 18. Wajah yang digunakan sebagai uji coba

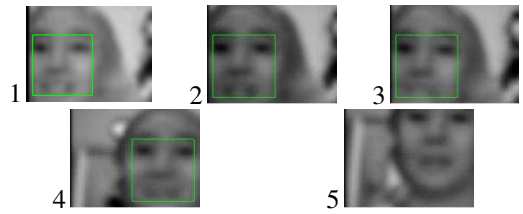
Tabel 10. Hasil uji coba dengan menggunakan arduino

No.	Skenario	*Detection Rate*
1	Skenario posisi wajah frontal dan miring	100%
2	Skenario mendeteksi koordinat wajah	80% - 100%
3	Skenario wajah yang bergerak	60% - 100%

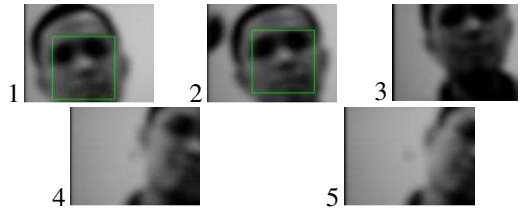
Uji kinerja dilakukan dengan cara menghitung kecepatan arduino dalam melakukan identifikasi wajah. Dimana, kecepatan tersebut dihitung mulai dari penangkapan obyek oleh serial kamera ov7670 *without FIFO* hingga arduino menjalankan algoritma deteksi wajah. Berdasarkan uji coba yang telah dilakukan, waktu yang dibutuhkan adalah sebesar 9-20 detik atau setara 0,1-0,05 fps dengan ukuran citra 40x30 piksel.



Gambar 19. Hasil deteksi wajah dengan skenario posisi wajah frontal dan miring



Gambar 20. Hasil deteksi wajah dengan skenario mendeteksi koordinat wajah



Gambar 21. Hasil deteksi wajah dengan skenario wajah yang bergerak

KESIMPULAN

Kesimpulan yang didapat dari penelitian ini adalah sebagai berikut:

1. Berdasarkan pada penelitian yang telah dilakukan, metode template matching merupakan metode yang paling cocok untuk diimplementasikan pada *embedded system* yang memiliki memori dan kecepatan yang terbatas.
2. Berdasarkan hasil *training* diketahui bahwa *template* wajah yang memiliki detection rate yang paling baik adalah template wajah yang terdiri dari mata dan hidung.
3. Berdasarkan hasil uji coba yang telah dilakukan dapat diketahui bahwa detection rate yang berhasil dicapai pada penelitian ini adalah sebesar 80% hingga 100%.

SARAN

Beberapa saran yang dapat dilakukan untuk menyempurnakan penelitian ini diantaranya adalah:

1. Untuk mempercepat proses pengambilan gambar, penggunaan serial kamera ov7670 without FIFO dapat diganti dengan serial kamera ov7670 with FIFO. Karena data gambar dari serial kamera ov7670 with FIFO bisa langsung didapatkan tanpa perlu memonitor kondisi VYSNC, HREF, dan PCLK.
2. Dari sisi kecepatan, sistem deteksi wajah yang diusulkan dalam penelitian ini belum cukup cepat. Sehingga perlu diuji coba dengan menggunakan perangkat keras yang memiliki kecepatan lebih cepat seperti arduino due.

DAFTAR PUSTAKA

[1] Banzi, M., *Getting Started with Arduino*. O'Reilly, 2011.

[2] Chen, C.R., Wong, W.S., dan Chiu, C.T., "A. 064 mm^2 Real-Time Cascade Face Detection Design Based on Reduced Two-Field Extraction". *IEEE, Transactions on Very Large Scale Integration (VLSI) Systems*, 19 (11), 2010.

[3] Colmenarez, A.J., dan Huang, T.S., Face Detection with Information-Based Maximum Disctimination. *Proc, IEEE Conf, Computer Vision and Pattern Recognition*, 1997, pp 782-787.

[4] Craw, I., Tock, D., dan Bennett, A., Finding Face Features. *Proc, Second European Conf, Computer Vision*, 1992, pp 92-96.

[5] Dai, Y., dan Nakano, Y., Face-Texture Model Based on SGLD and Its Application in Face Detection in a Color Scene. *Pattern Recognition*, 29(6), 1996, pp 1007-1017.

[6] Gomes, J.M., "Implementation of an intelligent sensor for measurement and prediction of solar radiation and atmospheric temperature". *IEEE, Intelligent Signal Processing (WISP)*. 2011.

[7] He, C., Papakonstantinou, A., dan Chen, D., A Novel SoC Architecture on FPGA for Ultra Fast Face Detection. *ICCD'09*, 4-7 Oct. 2009, pp. 412-418.

[8] Hori, Y., Shimizu, K., Nakamura, dan Y., Kuroda, T., "A Real-Time Multi Face Detection Technique Using Positive-Negative Lines-of-Face Template". *ICPR'04*, 1, 2004, pp. 765-768.

[9] Kamal, R., *Embedded Systems: Architecture, Programming and Design*. McGraw-Hill, 2008.

[10] Kjeldsen, R., dan Kender, J., Finding Skin in Color Images. *Proc, Second Int'l Conf, Automatics Face and Gesture Recognition*, 1996, pp. 312-317.

[11] Kyrkou, C., dan Theocharides, T., "A Flexible Parallel Hardware Architecture for AdaBoost-Based Real-Time Object Detection". VLSI, 19, 2010, pp. 1034-1047.

[12] Kyrkou, C., Ttofis, C., dan Theocharides, T., "FPGA-Accelerated Object Detection Using Edge Information". FPL, 2011, pp. 167-170.

[13] Lai, H.C., Savvides, dan M., Chen, T., "Proposed FPGA Hardware Architecture for High Frame Rate (>100 fps) Face Detection Using Feature Cascade Classifiers". *IEEE, Biometrics: Theory, Application, and Systems*, 2007.

[14] Lanitis, A., Taylor, C.J., dan Cootes, T.F., An Automatic Face Identification System Using Flexible Appearance Models. *Image and Vision Computing,* 13(5), 1995, pp. 393-401.

[15] Leung, T.K., Burl, M.C., dan Perona, P., Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching. *Proc, Fifth IEEE Int'l Conf, Computer Vision*, 1995. pp. 637-644.

[16] Lew, M.S., Information Theoretic View-Based and Modular Face Detection. *Proc, Second Intl'l Conf, Automatic Face and Gesture Recognition,* 1996. pp. 198-203.

[17] Li, S., Liu, W., Xin, D., dan Qiao, S., "An automatic identification tracking system applied to motion analysis of industrial field". *IEEE, Electric Information and Control Engineering*, 2011.

[18] McCready, R., Real-Time Face Detection on a Configurable Hardware System. *ACM, FPL '00,* 2000. pp. 157-162.

[19] McKenna, S., Gong, S., dan Raja, Y., Modelling Facial Colour and Identify with Gaussian Mixtures. Pattern Recognition, 31(12), 1998. pp. 1883-1892.

[20] Negru, S., "A conceptual architecture of an arduino-based social-emotional interactive system". *IEEE, Intelligent Computer Communication and Processing*. 2010.

[21] OmniVision, *OV7670/0V7171 CMOS VGA (640X480) CameraChip™ Implementation Guide*. OmniVision Technologies. 2005.

[22] OmniVision, *OV7670/OV7171 CMOS VGA (640X480) CameraChip™ with OmniPixel® Technology*. OmniVision Technologies. 2006.

[23] Osuna, E., Freund, R., dan Girosi, F., Training Support Vector Machines: An Application to Face Detection. *Proc, IEEE Conf, Computer Vision and Pattern Recognition,* 1997, pp. 130-136.

[24] Puspita, E., *Sistem Pendeteksi dan Penjejakan Wajah Secara Real Time*. Fakultas Teknologi Informasi. Institut Teknologi Sepuluh Nopember. 2004.

[25] Rajagopalan, A., Kumar, K., Karlekar, J., Manivasakan, R., Patil, M., Desai, U., Poonacha, P., dan Chaudhuri, S., Finding Faces in Photo graphs. *Proc, Sixth IEEE Intl'l Conf, Computer Vision,* 1988, pp. 640-645.

[26] Rowley, H., Baluja, S., dan Kanade, T., Neural Network-Based Face Detection. *IEEE Trans, Pattern Analysis and Machine Intelligence*, 20(1), 1998, pp. 23-38.

[27] Schneiderman, H., dan Kanade, T., Probabilistic Modeling of Local Appearance and Spatial Relationship for Object Recognition. *Proc, IEEE Conf, Computer Vision and Pattern Recognition,* 1998, pp. 45-51.

[28] Shalahuddin, M., Rosa, A.S., Belajar Pemrograman dengan Bahasa C++ dan Java dari Nol Menjadi Andal. Informatika Bandung, 2007.

[29] Sung, K.K., dan Poggio, T., Example-Based Learning for View-Based Human Face Detection. IEEE Trans, Pattern Analysis and Machine Intelligence, 20(1), 1998, pp. 39-51.

[30] Sutoyo, T., Mulyanto, E., Suhartono, V., Nurhayati, O.D., dan Wijanarto. *Teori Pengolahan Citra Digital.* ANDI. 2009.

[31] Theocharides, T., "A parallel Architecture for hardware face detection". *IEEE, Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, 2006.

[32] Turk, M., dan Pentland, A., Eigenfaces for Recognition. I.Cognitive Neuroscience, 3(1), 1991, pp. 71-86.

[33] Viola, P., Jones, M.J., "Robust Real-Time Face Detection". Kluwer Academic Publishers, International Journal of Computer Vision, 57(2), 2004, pp.137–154.

[34] Yang, G., dan Huang, T.S., Human Face Detection in Complex Background. Pattern Recognition. 27(1), 1994, pp. 53-63.

[35] Yang, J., dan Waibel, A., A Real-Time Face Tracker. *Proc, Third Workshop Applications of Computer Vision,* 1996, pp. 142-147.

[36] Yang, M.H., Kriegman, D., dan Ahuja, N., "Detecting faces in images: a survey". *IEEE, Transaction on Pattern, Analysis and Machine Intelligence,* 24(1), 2002, pp. 34-58.

[37] Yow, K.C., dan Cipolla, R., Feature-Based Human Face Detection. *Image and Vision Computing*, 15(9), 1997, pp. 713-735.

[38] Yutong, Z., Guosheng, dan Y., Licheng, W., "Fast Face in Field Programmable Gate Array". IEEE, *International Conference on Digital Manufacturing & Automation.* 2010.

[39] Zhang, X., Zhang, Y., dan Tian, Y., Performance Evaluation for Embedded System Based on Extension Theory. *IEEE, Knowledge Acquisition and Modeling,* 2008, pp. 389-391.