TESIS KI092361

SISTEM DETEKSI WAJAH PADA OPEN SOURCE PHYSICAL COMPUTING

YUPIT SUDIANTO
5110 202 002

DOSEN PEMBIMBING
Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.

PROGRAM MAGISTER
BIDANG KEAHLIAN SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2014

THESIS KI 092361

FACE DETECTION SYSTEM ON OPEN SOURCE PHYSICAL COMPUTING

YUPIT SUDIANTO
5110 202 002

SUPERVISOR
Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.

MAGISTER PROGRAM
MAJOR IN INFORMATION SYSTEM
DEPARTMENT OF INFORMATICS ENGINEERING
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2014

SISTEM DETEKSI WAJAH PADA OPEN SOURCE PHYSICAL COMPUTING

Nama : Yupit Sudianto
NRP : 5110 202 002
Pembimbing : Dr.Eng. Febriliyan Samopa, Skom., M.Kom

ABSTRAK

Deteksi wajah merupakan salah satu area penelitian yang menarik. Penelitian deteksi wajah yang dilakukan hingga saat ini mayoritas 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, karena komputer membutuhkan daya/watt yang besar.

Dalam penelitian ini diusulkan untuk membangun sistem deteksi wajah dengan menggunakan arduino. Sistem tersebut akan bersifat *autonomous* (mandiri), dengan kata lain peran komputer akan digantikan oleh arduino. Arduino yang digunakan adalah arduino Mega 2560 dengan spesifikasi mikrokontroler AT MEGA 2560, kecepatan 16 MHz, flash memory 256 KB, SRAM 8 KB. EEPROM 4 KB. Sehingga 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.

Detection rate yang berhasil dicapai dalam penelitian ini adalah sebesar 80%-100%. Dimana, keberhasilan dalam arduino dalam mengidentifikasi wajah dipengaruhi oleh jarak antara wajah manusia dengan kamera dan pergerakan manusia.

Kata Kunci : Arduino, Deteksi Wajah, *Embedded system*

Halaman ini sengaja dikosongkan

FACE DETECTION SYSTEM ON OPEN SOURCE PHYSICAL COMPUTING

Name	:	Yupit Sudianto
Student Identify Number	:	5110 202 002
Supervisor	:	Dr.Eng. Febriliyan Samopa, Skom., M.Kom

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

Halaman ini sengaja dikosongkan

SISTEM DETEKSI WAJAH PADA OPEN SOURCE PHYSICAL COMPUTING

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar Magister Komputer (M.Kom) di Institut Teknologi Sepuluh Nopember

oleh :

YUPIT SUDIANTO
NRP. 5110 202 002

Tanggal Ujian : 5 Februari 2014
Periode Wisuda : September 2014

Disetujui oleh:

1. **Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.**
   NIP. 19730219 199802 1 001 **(Pembimbing I)**

2. **Mahendrawathi ER, ST., M.Sc., Ph.D.**
   NIP. 19761011 200604 2 001 **(Penguji)**

3. **Erma Suryani, ST., MT., Ph.D.**
   NIP. 19700427 200501 2 001 **(Penguji)**

Direktur Program Pascasarjana,

Prof. Dr. Ir. Adi Soeprijanto, MT
NIP. 19640405 199002 1 001

KATA PENGANTAR

"Dengan menyebut nama Allah Yang Maha Pemurah lagi Maha Penyayang**"**

Segala puji hanya milik Allah Subhanahu wa Ta"aala. Shalawat dan salam semoga senantiasa dilimpahkan kepada Rasulullah shallallaahu „alaihi wa sallam, kepada keluarga, dan kepada para Shahabatnya.

Alhamdulillaah, dengan taufiq, pertolongan, dan rahmat Allah Subhanahu wa Ta"aala, penulis dapat menyelesaikan laporan penelitian yang bejudul "Sistem Deteksi Wajah Real Time pada *Open Source Physical Computing*" dengan baik. Laporan penelitian ini merupakan salah satu syarat kelulusan pada Magister Teknik Informatika, Bidang Keahlian Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Selama proses pengerjaan penelitian ini, penulis mendapatkan banyak bantuan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih yang sebesar-besarnya dan semoga Allah Subhanahu wa Ta"aala memberikan balasan kebaikan kepada:

- Bapak Dr. Eng Febriliyan Samopa, S.Kom., M.Kom. selaku Dosen Pembimbing yang bersedia meluangkan waktunya untuk membimbing penulis dalam menyelesaikan penelitian ini, serta Ibu Mahendrawathi ER, S.T., M.Sc., Ph.D dan Ibu Erma Suryani, S.T., M.T., Ph.D selaku penguji sidang penelitian.
- Bapak, Ibu, dan kedua kakakku yang terus mendorong penulis untuk segera menyelesaikan tesisnya.
- Teman-teman S2 Sistem informasi dan juga teman-temanku yang telah bersedia menjadi uji coba pada penelitian ini yaitu Danu Koeswara, Retno Aulia Vinarti, Fajar Annas, Aka, Riza, Rizki, Anif Bahwal, Zulmi, dan Heru.

Dan berbagai pihak yang tidak dapat penulis sebutkan satu per satu. Semoga Alloh Subhanahu wa Ta`ala membalas semua kebaikan yang telah dilakukan. Penulis menyadari bahwa masih banyak hal yang dapat dikembangkan pada tesis ini. Oleh karena itu, penulis menerima setiap saran dan kritik yang diberikan. Semoga tesis ini dapat memberikan manfaat.

Surabaya, Februari 2014

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
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 (McCready, 2000). Deteksi wajah memiliki peranan penting dalam berbagai bidang, seperti pengenalan wajah (*face recognition*), kamera pengawasan, pengenalan ekspresi, fotografi, dan lain sebagainya. Hingga saat ini, berbagai algoritma deteksi wajah telah diusulkan, dan umumnya algoritma tersebut diimplementasikan dengan menggunakan komputer atau laptop. Seiring dengan perkembangan teknologi yang menuju era *nanometer*, dan berkembangnya algoritma deteksi wajah, implementasi deteksi wajah pada *embedded system* bukanlah hal yang tidak mungkin untuk dilakukan.

1.1 Latar Belakang

Dalam beberapa tahun terakhir, deteksi wajah merupakan area penelitian yang menarik. 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. Karena sistem tersebut harus berjalan terus-menerus, maka akan memunculkan biaya per hari yang cukup besar yang harus dibayar oleh perusahaan. Salah satu cara yang dapat dilakukan untuk menghemat biaya adalah dengan menekan biaya pengadaan dan biaya operasional.

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). Menurut He dkk. (2009) 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. 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.

Arduino adalah *platform* dari *physical computing* yang bersifat *open source* (Banzi, 2011). 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.

Arduino telah digunakan pada beberapa penelitian. Li dkk. (2011) dengan papernya yang berjudul "*An Automatic Identification Tracking System Applied to Motion Analysis of Industrial Field*", dalam paper tersebut arduino digunakan untuk membangun sistem pelacakan identifikasi otomatis yang digunakan untuk analisa gerak pada bidang industri. Selain arduino, dalam penelitian tersebut juga menggunakan USB webcam dan potentiometer. Dengan mengontrol rotasi, operator dapat dilacak secara otomatis, dan rekaman pelacakan secara *real time* dapat tercatat. Gomes (2011) dengan judul "*Implementation of an intellegent sensor for measurement and prediction of solar radiation and atmospheric temperature*", paper tersebut mengembangkan sebuah sensor cerdas untuk memperoleh temperatur, data radiasi matahari, dan indeks perkiraan curah hujan. Data tersebut digunakan untuk memperkirakan temperatur dan radiasi matahari di masa datang. Penelitian tersebut menggunakan arduino yang dilengkapi dengan sensor temperature dan radiasi matahari. Negru (2010) dengan judul "*A conceptual Architecture of an Arduino-based Social Emotional Interactive System*", paper tersebut menyajikan *platform* perangkat lunak untuk mengendalikan robot hexapod dengan menggunakan Matlab. Arduino pada penelitian tersebut digunakan untuk mengendalikan bagian kaki robot hexapod.

Kebutuhan daya yang diperlukan oleh arduino Mega 2560 dipengaruhi oleh besar kecilnya tegangan dan arus listrik yang digunakan. Tegangan yang

direkomendasikan pada arduino Mega 2560 adalah sebesar 7-12 Volt dan arus DC per pin I/O (54 pin digital 16 pin analog) sebesar 50 mA. Untuk menghitung kebutuhan daya pada arduino digunakan rumus sebagai berikut:

$$\text{daya (Watt)} = \text{tegangan listrik (Volt)} \times \text{ kuat arus listrik (Ampere)} \quad (1.1)$$

Sistem deteksi wajah yang diimplementasikan pada arduino Mega 2560 akan berinteraksi dengan kamera *ov7670 without FIFO*, 2 buah servo, dan 2 lampu led. Untuk komunikasi dengan kamera diperlukan 13 pin, sedangkan komunikasi dengan 2 buah servo diperlukan 2 pin, dan untuk komunikasi dengan 2 buah lampu led diperlukan 2 pin, sehingga total pin yang diperlukan adalah 17 pin. Berdasarkan rumus 1.1, jika tegangan yang digunakan oleh arduino Mega 2560 sebesar 7.5 volt, maka daya yang dibutuhkan adalah sebesar 6,375 Watt. Sehingga sistem deteksi wajah yang diimplementasikan pada arduino Mega 2650 akan lebih menghemat daya dari pada sistem deteksi wajah yang dibangun dengan menggunakan komputer.

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, Hori dkk. (2004) berhasil mencapai kecepatan sebesar 30 fps dengan ukuran gambar sebesar 320x240 piksel. Penelitian yang dilakukan oleh Hori dkk. (2004) menggunakan metode yang berdasarkan pada warna kulit dan *lines-of-face template*. He dkk. (2009) menggunakan algoritma ANN, kecepatan yang berhasil dicapai pada penelitian ini sebesar 625 fps dengan ukuran gambar sebesar 640x480 piksel. Kyrkou dkk. (2010) dengan menggunakan algoritma AdaBoost, kecepatan yang berhasil dicapai adalah 64-139 fps dengan gambar sebesar 1024x768 piksel. Yutong dkk. (2010) mengusulkan metode yang berdasarkan warna kulit, dan kecepatan yang berhasil dicapai adalah sebesar 60 fps dengan ukuran gambar sebesar 1280x1024 piksel. Kyrkou dkk. (2011) menggunakan *edge information,* dan kecepatan yang berhasil dicapai adalah 28-60 fps untuk gambar dengan ukuran 320x240 piksel. Dari berbagai penelitian tersebut akan dipilih algoritma mana yang paling sesuai dengan arduino. Penelitian yang dilakukan oleh Yang dkk. (2002) mengenai karakteristik metode-metode deteksi

wajah, akan dijadikan sebagai dasar untuk memilih algoritma yang akan diimplementasikan pada arduino.

Menurut Yang dkk. (2002) secara umum metode deteksi wajah dikelompokan dalam empat kategori, yaitu metode *knowledge-based*, metode *feature-invariant*, metode *template-matching*, dan metode *appearance-based*. Metode *knowledge-based* menggunakan hubungan antara fitur wajah untuk mengidentifikasi keberadaan wajah. Contoh fitur wajah yang digunakan adalah keberadaan dua mata yang simetris, posisi hidung di tengah dan posisi mulut di bawah hidung. Kelemahan dari metode ini adalah semakin ketat aturan yang digunakan, kemungkinan terjadinya kegagalan dalam deteksi wajah juga besar. Sebaliknya, jika aturan yang digunakan terlalu umum, maka *false positive* (kegagalan menolak yang bukan wajah) yang dihasilkan akan semakin besar. Untuk metode *feature-invariant* pendekatan yang digunakan adalah dengan cara menemukan dan menentukan fitur-fitur umum yang terdapat pada wajah, bahkan ketika wajah dalam berbagai posisi, kondisi pencahayaan yang berbeda, dan kondisi lainnya. Fitur yang umum digunakan dalam metode ini adalah warna kulit dan *edge*. Kelemahan yang dimiliki metode ini sama seperti metode *knowledge-based*. Sedangkan metode *template-matching* mengidentifikasi wajah dengan menggunakan pola wajah yang telah ditentukan. Metode ini menghitung korelasi antara gambar dengan pola yang disimpan. Kelemahan dari metode ini adalah kesulitan dalam mendapatkan *template* wajah yang umum. Berbeda dengan metode-metode sebelumnya, metode *appearance-based* menggunakan analisis statistik dan *machine learning* untuk mengumpulkan informasi dari satu set *training*. Kelemahan dari metode ini adalah memerlukan waktu yang lama untuk melakukan proses *training* (Yang dkk., 2002).

Pada penelitian ini akan dibangun sistem deteksi wajah yang bersifat *autonomous* dengan menggunakan arduino, sehingga tidak perlu menggunakan peran komputer dalam melakukan proses komputasi untuk mengidentifikasi wajah. Selain mendeteksi wajah, sistem juga dapat mengikuti pergerakan wajah manusia, jika wajah tersebut bergerak ke kiri atau ke kanan maka kamera akan mengikuti pergerakan wajah tersebut dengan menempatkan posisi wajah pada tengah kamera. Jika ada sekumpulan manusia, maka yang terdeteksi adalah

manusia yang berada pada posisi paling depan. Pembangunan sistem deteksi wajah pada arduino sejauh ini masih memanfaatkan peran komputer. Komputer digunakan untuk menjalankan algoritma deteksi wajah (dengan menggunakan *library* OpenCV), sedangkan arduino dimanfaatkan untuk menggerakkan kamera mengikuti gerakan wajah. Sehingga sistem yang dibangun belum bersifat *autonomous* (mandiri).

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. Sehingga harus dilakukan penelitian untuk mendapatkan algoritma deteksi wajah yang sesuai dengan kemampuan arduino Mega 2560.

1.2 Perumusan Masalah

Beberapa penelitian deteksi wajah telah mengusulkan penggunaan *embedded system* untuk mengatasi besarnya kebutuhan daya listrik. Perangkat keras yang digunakan dalam penelitian tersebut adalah FPGA (*Field Programmable Gate Arrays*). FPGA merupakan perangkat keras untuk membuat *prototype* dari sistem yang ingin dibangun. Hasil rancangan sistem dari FPGA akan diberikan ke pabrik untuk dibangun menjadi sistem yang sesuai dengan rancangan.

Untuk membangun sistem deteksi wajah yang *autonomous*, pada penelitian ini diusulkan penggunaan arduino Mega 2560 sebagai perangkat keras dalam pembangunan sistem deteksi wajah. Berbeda dengan komputer, arduino UNO memiliki keterbatasan memori (*flash memory* 256 KB RAM dengan 8 KB yang digunakan oleh *bootloader*, 8 KB dari SRAM, dan 4 KB EEPROM) dan kecepatan prosesor (sebesar 16 MHz), sehingga perlu diadakan penelitian mengenai:

1. Algoritma deteksi wajah yang sesuai dengan karakteristik arduino Mega 2560. Karena tidak semua algoritma deteksi wajah yang ada dapat diimplementasikan pada arduino Mega 2560, sehingga harus dicari algoritma deteksi wajah yang sesuai dengan karakteristik arduino.
2. Bagaimana pengimplementasian algoritma deteksi wajah yang dipilih, agar dapat berjalan dengan maksimal pada arduino.

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini adalah mendapatkan algoritma untuk mendeteksi wajah yang dapat diimplementasikan pada arduino.

Manfaat dari penelitian ini adalah:

1. Didapatkan algoritma pendeteksi wajah yang sesuai dengan kemampuan arduino.
2. Memberikan opini baru bagi peneliti untuk mengembangkan metode-metode deteksi wajah pada arduino ataupun *embedded system* yang lain.

1.4 Batasan Penelitian

Batasan dari penelitian ini adalah:

1. Penelitian ini dititik beratkan pada deteksi wajah, artinya mencari bagian dari gambar yang merupakan wajah.
2. Batasan real time yang digunakan pada penelitian ini, berdasarkan pada kemampuan dari kamera yang digunakan. Kamera yang digunakan adalah ov7670 *without FIFO*.
3. Pergerakan orang tidak terlalu cepat, hal ini dipengaruhi oleh mutu dari kamera yang digunakan.
4. Faktor cahaya dalam penelitian ini diasumsikan berada pada tingkat intensitas yang cukup dalam aktivitas sosial. Jika kemampuan kamera yang digunakan kurang, maka dapat ditambahkan lampu penerangan.
5. Penelitian ini menggunakan arduino Mega 2560 yang memiliki kecepatan 16MHz, *flash memory* 256 Kb, RAM 8kb, dan EEPROM 4 Kb.

1.5 Kontribusi Penelitian

Penggunaan arduino dalam pembangunan sistem deteksi wajah belum pernah dilakukan sebelumnya. Namun, bukan berarti arduino tidak dapat digunakan untuk melakukan deteksi wajah, karena pada dasarnya arduino adalah komputer mini dengan keterbatasan prosesor dan memory. Perbedaan arduino dengan komputer adalah besarnya memory dan kecepatan yang dimiliki. Sehingga kontribusi utama dalam penelitian ini adalah mendapatkan algoritma deteksi wajah yang paling sesuai untuk diimplementasikan pada arduino.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

Pada bab ini dijelaskan mengenai teori-teori yang digunakan dalam penyusunan tesis, yaitu pengenalan deteksi wajah, deteksi wajah pada *embedded system*, perangkat keras *embedded system* yang digunakan, pengenalan algoritma, dan *histogram equalization*.

2.1 Pengenalan Deteksi Wajah

Deteksi wajah merupakan area penelitian yang menantang dan menarik dalam beberapa tahun terakhir. Teknik deteksi wajah ini telah digunakan dalam berbagai bidang penelitian antara lain pengenalan wajah, kamera pengawasan, pengenalan ekspresi, fotografi, dan lain-lain.

Menurut McCready (2000) deteksi wajah adalah 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. Tantangan lain yang juga perlu diperhatikan adalah posisi wajah (frontal, *in-plane rotation dan out-of-plane rotation*) dan skala ditunjukkan dalam gambar 2.1. Variasi ini harus ditangani dengan baik oleh algoritma deteksi wajah, ketika akan diimplementasikan pada lingkungan yang tidak terkendali.

Yang dkk. (2002) mengelompokkan teknik deteksi wajah menjadi 4 kategori yaitu metode *knowledge-based*, metode *feature-invariant*, metode *template-matching*, dan metode *appearance-based*. Berikut ini merupakan penjelasan dari masing-masing metode tersebut:

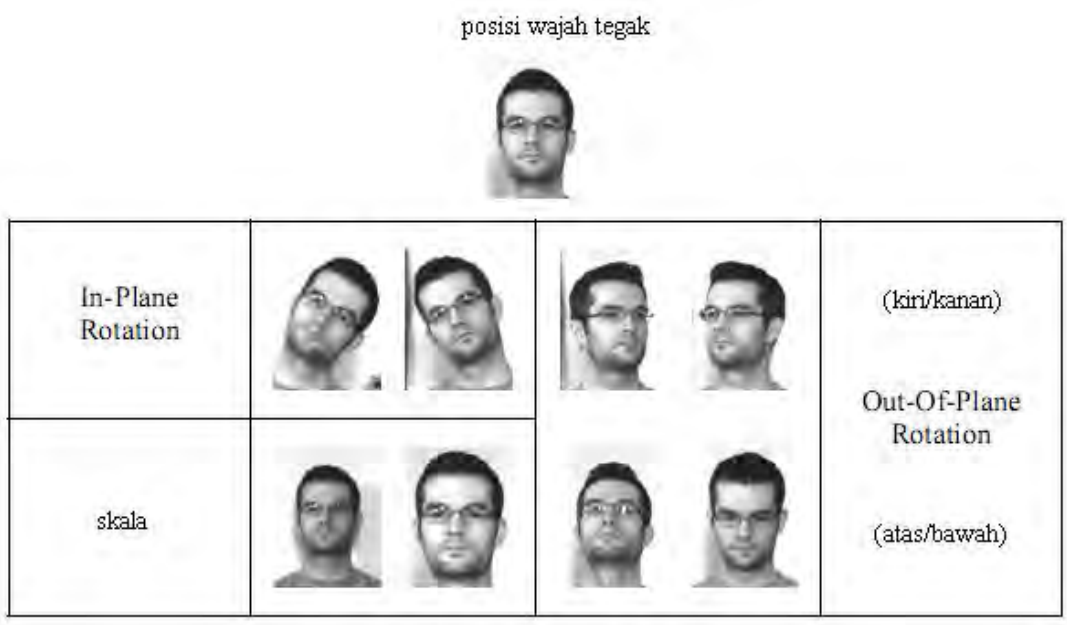
- **Metode *knowledge based***

  Metode *knowledge based* menggunakan hubungan antara fitur wajah untuk mengidentifikasi keberadaan wajah. Contoh fitur wajah yang digunakan adalah keberadaan dua mata yang simetris, posisi hidung di tengah dan posisi

mulut di bawah hidung. Kelemahan dari metode ini adalah semakin ketat aturan yang digunakan, kemungkinan terjadinya kegagalan dalam deteksi wajah juga besar. Sebaliknya, jika aturan yang digunakan terlalu umum, maka false positive (kegagalan menolak yang bukan wajah) yang dihasilkan akan semakin besar.

- **Metode *feature invariant***

  Metode *feature invariant* menggunakan pendekatan dengan cara menentukan fitur-fitur umum yang terdapat pada wajah, bahkan ketika wajah dalam berbagai posisi, kondisi pencahayaan yang berbeda, dan kondisi lainnya. Fitur yang umum digunakan dalam metode ini adalah warna kulit, garis dan bentuk. Kelemahan yang dimiliki metode ini sama seperti metode *knowledge-based*. Karena kesamaan tersebut Tsao dkk. (2010), menggabungkan metode *knowledge based* dan metode *feature invariant* dalam satu kategori.



Gambar 2.1 Variasi Posisi Wajah (McCready, 2000)

- **Metode *template matching***

  Metode *template matching* mengidentifikasi wajah dengan menggunakan pola wajah yang telah ditentukan. Proses identifikasi wajah pada metode ini dilakukan dengan cara menghitung korelasi antara citra dengan pola yang

disimpan. Kelemahan dari metode ini adalah kesulitan dalam hal menentukan *template* wajah yang umum.

- **Metode *appearance based***

  Metode *appearance based* menggunakan analisis statistik dan *machine learning* untuk mengumpulkan informasi dari satu set *training*. Kelemahan dari metode ini adalah memerlukan waktu yang lama untuk melakukan proses *training*.

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 cara:

$$\text{detection rate} = \frac{\text{jumlah wajah yang berhasil dideteksi}}{\text{jumlah wajah yang diujikan}} \text{x } 100\% \qquad (2.1)$$

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

$$\text{false positive} = \frac{\text{jumlah bukan wajah yang dianggap wajah}}{\text{jumlah total window yang diuji}} \text{x } 100\% \qquad (2.2)$$

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

$$kecepatan\ deteksi = \frac{frames}{waktu\ saat\ ini - waktu\ mulai} \qquad (2.3)$$

2.2 Deteksi Wajah pada Embedded System

Berbagai penelitian deteksi wajah yang berbasiskan *embedded system* telah banyak dilakukan. Dari penelitian yang ada, dipilih yang berhasil mencapai kecepatan deteksi yang cukup tinggi. Jika dikelompokkan dalam kategori deteksi wajah yang diusulkan oleh Yang dkk. (2002), terbagi dalam 3 metode, yaitu metode *feature invariant*¸ metode *template matching*, dan metode *appearance based*.

2.2.1 Metode Feature Invariant

Terdapat dua penelitian yang menggunakan metode *feature invariant*, yaitu yang dilakukan oleh Yutong dkk. (2010) dengan menggunakan warna kulit dan penelitian yang dilakukan oleh Kyrkou dkk. (2011) yang menggunakan *edge information*. Berikut ini penjelasan singkat dari penelitian tersebut:

- **Berdasarkan warna kulit**

  Secara umum cara kerja algoritma deteksi wajah yang berdasarkan warna kulit adalah dengan cara mengelompokkan karakteristik warna kulit, kemudian mengeluarkan sebanyak mungkin komponen warna latar belakang dan melakukan segmentasi gambar. Penelitian deteksi wajah pada *embedded system* yang berdasarkan warna kulit telah dilakukan oleh Yutong dkk. (2010) dalam papernya yang berjudul "*Fast Face Detection in Field Programmable Gate Array*". Dalam penelitian tersebut, Yutong dkk. (2010) mengusulkan 3 tahap yaitu pemilihan ruang warna, binerisasi, dan komputasi *fixed-point.* Tiga tahap tersebut dijelaskan sebagai berikut:

  1. Pemilihan ruang warna

     Pada tahap ini dilakukan perubahan format gambar menjadi RGB. Penggunaan format RGB dapat meningkatkan kecepatan dan menghemat kapasitas penyimpanan. Dalam tahap ini dilakukan normalisasi pada RGB menjadi NCC (*Normalized Color Coordination*), untuk mengatasi kepekaan ruang warna RGB terhadap kecerahan pencahayaan (*brightness*).

  2. Binerisasi

     Dalam tahap ini akan dilakukan pengelompokan gambar menjadi dua kategori yaitu, wilayah wajah dan bukan wajah. Tahap binerisasi digunakan untuk menyederhanakan gambar sehingga akan memudahkan tahap berikutnya.

  3. Komputasi *fixed-point*

     Penelitian ini mengusulkan penggunaan operasi *fixed-point* untuk menghemat waktu dan kapasitas penyimpanan.

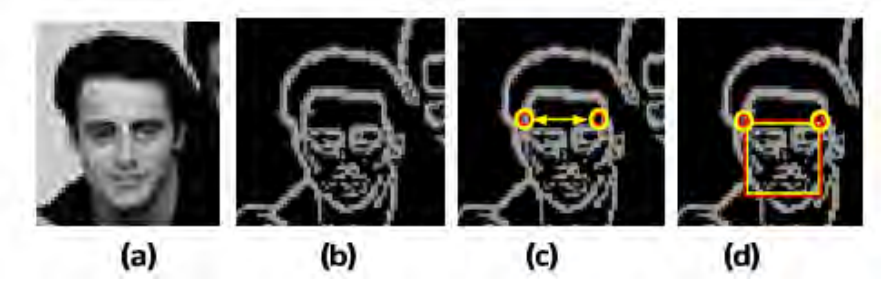
Penelitian ini menggunakan perangkat keras Altera EP2C70896C6 FPGA yang digabungkan dengan chip 32MB SDRAM. Dalam penelitian ini kecepatan yang berhasil dicapai adalah 60 fps dengan ukuran gambar 1280x1024 piksel.

- **Berdasarkan Edge Information**

  Kyrkou dkk. (2011) menggunakan *edge information* untuk mengurangi ruang pencarian ketika mengidentifikasi wajah. Dengan menggunakan perangkat keras Virtex-5 LX110T FPGA kecepatan yang berhasil dicapai dalam penelitian tersebut sebesar 28-60 fps.

  Proses deteksi wajah dengan menggunakan *edge information* ditunjukan pada gambar 2.2 (Kyrkou dkk., 2011). *Edge information* digunakan untuk mengabaikan daerah yang tidak menjanjikan (daerah gambar yang tidak menunjukkan perubahan piksel yang tinggi, sehingga daerah tersebut dianggap tidak mengandung informasi yang dibutuhkan).

  Edge menunjukkan skema wajah, sehingga dapat diperoleh ukuran dari wajah (gambar 2.2 bagian c). Sehingga *edge* dapat digunakan untuk membangun *window* disekitar objek, dengan cara mencari *edge* yang membentuk persegi (gambar 2.2 bagian d). Window yang terbentuk akan ditentukan apakah termasuk latar belakang atau bukan. Selanjutnya algoritma klasifikasi difokuskan pada wilayah yang bukan latar belakang.



Gambar 2.2 Empat Tahap Deteksi Wajah dengan Menggunakan Edge Information. (a) gambar masukan (b) menghitung edge image (c) menemukan edge yang pertama dan kedua dan menghitung jaraknya (d) menggunakan jarak tersebut untuk membentuk window. (Kyrkou dkk., 2011)

2.2.2 Metode Template Matching

Sistem deteksi wajah dengan menggunakan metode *template matching* diusulkan oleh Hori dkk. (2004). Dalam penelitian tersebut Hori dkk. (2004) berhasil mencapai kecepatan sebesar 30 fps dengan ukuran gambar 320x240 piksel. Berikut ini penjelasan singkat dari penelitian tersebut:

- **Berdasarkan Positive-Negative Lines-of-Face Template**

  Dalam penelitian yang dilakukan oleh Hori dkk. (2004) kecepatan yang berhasil dicapai adalah sebesar 30 fps untuk gambar yang berisikan 6 wajah dengan ukuran 320x240 piksel. Algoritma yang digunakan dalam penelitian ini adalah yang berdasarkan warna kulit dan *lines-of-face template*. Ada 3 tahap dalam penelitian ini (ditunjukkan pada Gambar 2.3). Tiga tahap tersebut adalah (i) sebelum pemrosesan gambar untuk mendapatkan tepi wajah, (ii) mendeteksi garis wajah dengan menggunakan SSGA (*Steady State Genetic Algorithm*), dan (iii) menetapkan wajah dengan mencari daerah mulut. Tahap pertama dan kedua diimplementasikan pada FPGA dengan *logic* sebesar 40k *gates* dan memori 249k *gates*. Sedangkan tahap ketiga diimplementasikan pada mikrokontroler HITACHI SH-3. Gambar 2.5 menunjukkan hasil deteksi wajah. Berikut ini adalah penjelasan dari ketiga tahap tersebut:

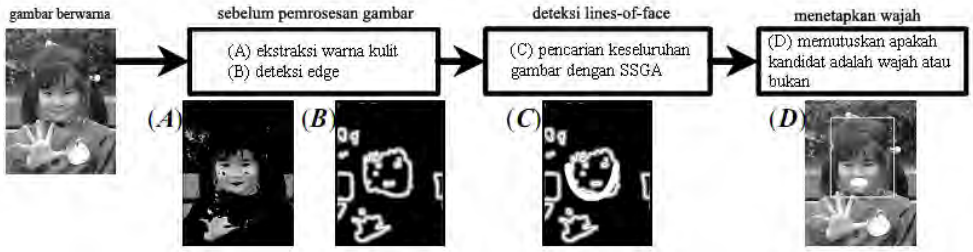
  i. Sebelum pemrosesan citra

     Tahap ini terbagi menjadi 2 bagian. Bagian pertama menentukan model warna kulit yang sesuai agar proses ekstrasi warna kulit dapat berjalan dengan efektif. Model warna kulit yang diusulkan pada penelitian ini dilakukan dengan mempertimbangkan karakteristik warna dari sistem masukan gambar. Bagian yang kedua, setelah diperoleh gambar dengan ekstraksi warna kulit (gambar 2.3 bagian A), dilakukan deteksi tepi warna kulit dan pengkaburan gambar. Untuk melakukan ekstraksi tepi warna kulit (gambar 2.3 bagian B) digunakan *Laplacian filter*.

  ii. Mendeteksi wajah dengan menggunakan SSGA

     Pada tahap ini menjelaskan tahapan deteksi tepi wajah dari gambar dengan menggunakan *positive-negative lines-of-face template* dan

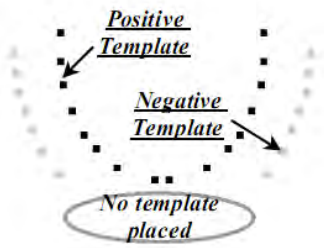
SSGA sebagai prosedur pencarian global (gambar 2.3 bagian C). Pada gambar 2.4, diusulkan sebanyak 30 poin untuk menyusun *lines-of-face template*. Poin yang berwarna hitam merupakan *positive template* yang menandakan keberadaan garis wajah. Sedangkan poin yang berwarna abu-abu merupakan *negative template* yang menandakan tidak ada garis wajah.



Gambar 2.3 Tiga Tahap Deteksi Wajah dengan Menggunakan *Positive-Negative Lines-of-Face Template* (Hori dkk., 2004)

iii. Menetapkan wajah dengan mencari daerah mulut

Penggunaan *positive-negative template* memiliki kemungkinan untuk mendeteksi *false positive*. Sehingga dilakukan pendekatan dengan deteksi mulut di area yang terdeteksi (gambar 2.3 bagian D). Wilayah bibir diperoleh dengan mencari wilayah yang memiliki ambang batas warna tertinggi dari area yang terdeteksi.



Gambar 2.4 Positive-negatice lines-of-face template (Hori dkk, 2004)

2.2.3 Metode Appearance Based

Metode *appearance based* diusulkan oleh Kyrkou dkk. (2010) dan He dkk. (2009). Kyrkou dkk. (2010) menggunakan algoritma AdaBoost sedangkan He dkk. (2009) menggunakan algoritma ANN (*artificial neural network*). Berikut ini penjelasan singkat dari penelitian tersebut:

- **Berdasarkan AdaBoost**

  Penggunaan Algoritma AdaBoost untuk sistem deteksi wajah pada *embedded system* diusulkan oleh Kyrkou dkk. (2010). Kecepatan yang berhasil dicapai adalah sebesar 64-139 fps dengan ukuran gambar 1024x768 piksel. Penelitian tersebut menggunakan perangkat keras Xilinx Virtex II Pro FPGA.

Gambar 2.5 Hasil deteksi wajah dengan menggunakan *Positive-Negative Lines-of-Face Template* (Hori dkk., 2004)

Kyrkou dkk. (2010) dalam papernya yang berjudul "A Flexible Parallel Hardware Architecture for AdaBoost-Based Real-Time Object Detection" menggunakan algoritma AdaBoost untuk membangun sistem deteksi wajah. Algoritma AdaBoost yang digunakan oleh Kyrkou dkk. (2010) mengacu pada penelitian sebelumnya yang dilakukan oleh Viola dkk. (2004). Berdasarkan penelitian Viola dkk. (2004) ada tiga kontribusi utama yang mendukung pencapaian sistem deteksi wajah yang cepat, yaitu menyatakan gambar dalam bentuk *integral image*, membuat algoritma pelatihan dengan menggunakan AdaBoost, dan membuat sistem deteksi wajah dalam bentuk *cascade classifier*.

Classifier terbentuk dari selisih sekelompok piksel dalam suatu daerah tertentu. Selisih sekelompok piksel didapatkan dari pengurangan sekelompok piksel dalam daerah berwarna putih dengan sekelompok piksel dalam daerah berwarna hitam. Ada 3 macam bentuk yang digunakan, yaitu selisih dari dua kelompok piksel (gambar 2.6 bagian A dan B), selisih dari tiga kelompok piksel (gambar 2.6 bagian C), dan selisih dari empat kelompok piksel (gambar 2.6 bagian D).

Integral image digunakan untuk menyatakan bidang gambar dalam bentuk nilai integral dua dimensi. Dengan menggunakan *integral image* pada suatu gambar akan lebih mudah untuk mendapatkan nilai dari sekelompok piksel dan memiliki jumlah komputasi yang tetap (sekelompok piksel dengan ukuran 10x10 piksel maupun 100x100 piksel membutuhkan waktu komputasi yang sama). Kelebihan lain dari *integral image* adalah pada saat proses penskalaan (*scaling*), dimana proses tersebut tidak diperlukan lagi. Dengan mengambil satu formulasi perhitungan, akan diperoleh kemudahan dalam membaca piksel sesuai dengan ukuran dan skala yang diinginkan. Gambar *integral image* ditunjukkan pada gambar 2.7.

Nilai piksel dari *integral image* adalah nilai total piksel dari citra asli pada koordinat (0,0) sampai (x,y), yaitu,

$$ii(x,y) = \sum_{b=0}^{y} \sum_{k=0}^{x} i(k,b) \quad (2.4)$$

Dimana,
ii(x,y) : nilai piksel dari integral image pada (x,y)
i(b,k) : nilai piksel dari citra asli pada (k,b)

Sedangkan nilai piksel dari bidang kotak tertentu (seperti pada gambar 2.8) pada gambar asli dapat dihitung dengan,

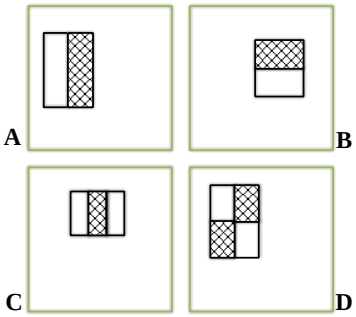
$$L = \sum_{b=y1}^{y2} \sum_{k=x1}^{x2} i(k,b), \text{ atau} \quad (2.2)$$

$$L = ii(x2,y2) + ii(x1,y1) - ii(x2,y1) - ii(x1,y2) \quad (2.3)$$

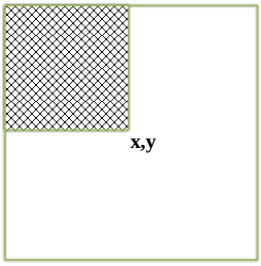
Dimana,
L : Nilai penjumlahan piksel pada citra di koordinat (x1,y1)-(x2,y2)

Sedangkan AdaBoost merupakan algoritma pemilihan dan pembelajaran (pelatihan) yang digunakan untuk meningkatkan kinerja dari sekelompok *classifier* yang sederhana sehingga menjadi *classifier* yang kuat. AdaBoost bekerja berdasarkan pemilihan dari sekelompok *classifier* (ciri-ciri sederhana dari wajah), dimana diantara *classifier* tersebut memiliki kesalahan yang

paling rendah. Pemilihan *classifier* dilakukan terus menerus hingga mendapatkan sejumlah *classifier* yang sederhana, sehingga dapat dibentuk *classifier* yang lebih kuat dari sekelompok *classifier* yang telah terpilih. Selain melakukan pemilihan *classifier*, AdaBoost juga melakukan pelatihan, yang dilakukan dengan cara menentukan faktor penguat dari masing-masing *classifier* yang terpilih.



Gambar 2.6 *Classifier*



Gambar 2.7 *Integral image*

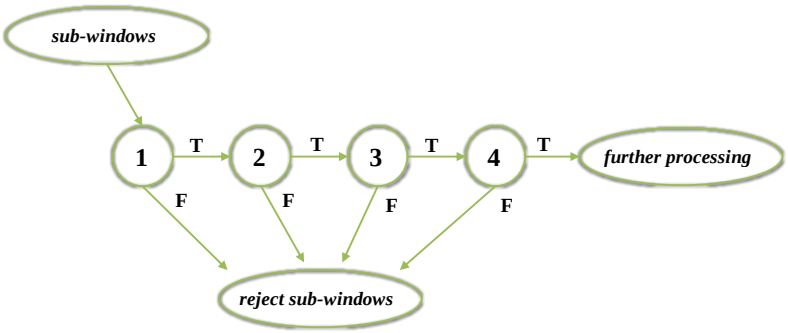
x1,y1		x2,y1
x1,y2		x2,y2

Gambar 2.8 Perhitungan jumlah piksel dalam daerah tertentu

Cascade classifier adalah suatu cara untuk membagi suatu *classifier* yang kuat tetapi lambat menjadi sekelompok *classifier* yang lebih kecil dan disusun dengan cara bertingkat (*cascade*). Jika ada sebuah *classifier* kuat yang terdiri dari gabungan 200 ciri sederhana dapat dibagi menjadi 10 *classifier* dalam bentuk *cascade*, dimana masing-masing terdiri dari kombinasi 20 ciri sederhana. Struktur *cascade classifier* ditunjukkan pada gambar 2.9

Penelitian yang dilakukan oleh Kyrkou dkk. (2010) mengusulkan lima isu utama. Lima isu utama tersebut adalah skala gambar, komputasi *integral image*, komputasi fitur, tahap komputasi dan identifikasi wilayah yang berisi wajah. Kelima isu utama tesebut tercakup dalam 2 blok utama, yaitu unit *image pyramid generation* (IPG) dan *systolic array*.

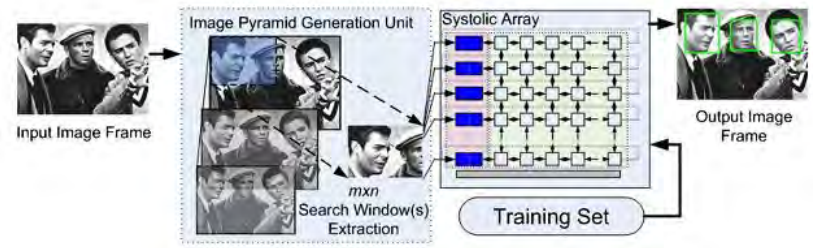
IPG menerima masukan frame video dan menghasilkan daerah gambar yang diproses *systolic array*. *Systolic array* mengevaluasi wilayah kandidat yang memiliki potensi terdapat wajah. Dalam penelitian ini digunakan *hybrid scaling mechanism* dengan memanfaatkan masukan gambar *downscaling*, serta skema fitur *upscaling*, seperti yang diusulkan oleh Viola dan Jones (2004). Fitur *upscaling* terus beriterasi, untuk mendeteksi wajah dalam *window*. Ukuran window lebih besar dari pada ukuran fitur. Iterasi terus berlanjut hingga ukuran fitur sama dengan ukuran *window* (Fitur berukuran persegi, sedangkan *window* dapat berupa persegi panjang).



Gambar 2.9 *Cascade classifier*

IPG memproses gambar input dan menghasilkan *window*. Setiap *window* kemudian diproses secara paralel, fitur dengan fitur dan tahap dengan tahap melalui *systolic array*. Array bertanggung jawab untuk menghitung *integral image*, menghitung persegi panjang untuk setiap fitur, dan mengevaluasi kedua fitur dan jumlah tahap. Jika wajah dideteksi dalam *window*, array mengeluarkan koordinat dari wilayah yang berisi wajah, dengan mempertimbangkan skala dari fitur bahwa wajah telah ditemukan. Ketika array telah menyelesaikan pemeriksaan *window*, *window* baru dimasukkan ke

dalam array dari IPG, hingga keseluruhan gambar telah dicari. Blok diagram dari sistem arsitektur ditunjukkan pada gambar 2.10.



Gambar 2.10 Blok diagram sistem arsitektur deteksi wajah dengan AdaBoost (Kyrkou dkk., 2010)

- **Berdasarkan ANN (*Artificial Neural Network*)**

  He dkk. (2009) mengusulkan penggunaan ANN untuk sistem deteksi wajah pada *embedded system.* Perangkat keras yang digunakan adalah Xilinx FX 130T Virtex5 FPGA. Dalam penelitian tersebut kecepatan yang berhasil dicapai adalah sebesar 625 fps.

  Penelitian yang dilakukan oleh He dkk. (2009) mengusulkan tiga tahap, yaitu tahap sebelum pemrosesan, tahap deteksi, dan tahap setelah pemrosesan. Berikut ini penjelasan mengenai ketiga tahap tersebut:

  - Tahap sebelum pemrosesan

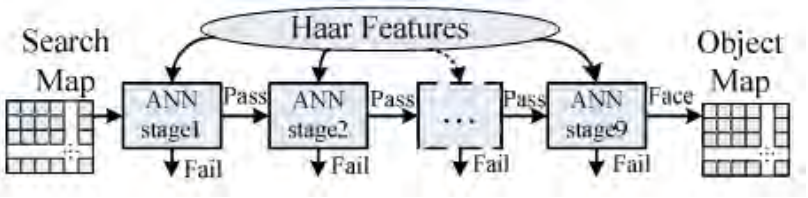
    Dalam tahap ini dilakukan manipulasi terhadap gambar, dengan tujuan untuk mengurangi beban kerja pada tahap deteksi. Manipulasi gambar dilakukan dengan cara mengubah ukuran gambar menjadi 80x60 piksel. Kemudian dibentuk frame grayscale dengan intensitas 8 bit, yang dihasilkan dari frame warna 80x60. Frame grayscale digunakan dalam deteksi gerak, sedangkan frame warna digunakan untuk deteksi kulit. Deteksi gerak dan deteksi kulit digunakan untuk menyaring wilayah gambar yang memiliki kemungkinan terdapat wajah.

  - Tahap deteksi

    Pada tahap deteksi dihasilkan *integral image*, dengan tujuan untuk meningkatkan komputasi *haar feature*. Gambar *grayscale* 80x60 digunakan untuk membuat *integral image*. Tiga ukuran objek digunakan untuk memindai *window* dari *integral image*, yaitu 11x11,

19x19, dan 27x27 (yang sesuai dengan ukuran objek 88x88, 152x152, dan 216x216 dalam frame asli 640x480). Ketika pemindaian *window* sudah sesuai dengan pengaturan bit dalam peta pencarian, maka proses klasifikasi akan didorong untuk menentukan apakah terdapat keberadaan wajah dalam *window*.

Proses klasifikasi dilakukan dengan 9 tahap cascade dari ANN. Setiap tahap ANN mengambil masukan dari serangkaian *haar feature* yang telah dihitung dengan *integral image*, dan hanya diaktifkan jika tahap ANN sebelumnya menghasilkan keluaran yang positif. Gambar 2.11 menunjukkan struktur dari 9 tahap *cascade*.



Gambar 2.11 Sembilan Tahap Cascade Classifier ANN (He dkk., 2009)

- Tahap setelah pemrosesan

  Tahap ini bertujuan untuk menghilangkan gangguan yang terdeteksi dan tumpang tindih artifak (yakni tumpang tindih deteksi pada wajah yang sama). Ketika wilayah wajah diidentifikasi dalam peta objek, dilakukan pengecekan tumpang tindih dengan penemuan wilayah wajah yang sebelumnya telah dilakukan. Jika tidak ditemukan tumpang tindih, lokasi *window* objek yang sesuai ditambahkan pada daftar wilayah wajah, dan bit peta objek yang dicakupkan pada *window* objek di bersihkan. Identifikasi lokasi wajah yang baru dilanjutkan hingga tidak ada wilayah persegi 5x5 pada salah satu peta objek yang memiliki jumlah elemen lebih besar dari *threshold* yang telah ditetapkan. Deteksi wajah pada frame selesai ketika hal ini terjadi.

2.3 Embedded System

Embedded system adalah sistem yang memiliki perangkat lunak yang tertanam pada perangkat keras, dan sistem tersebut merupakan bagian dari

aplikasi, atau bagian khusus dari aplikasi maupun produk, atau bahkan dari sistem yang lebih besar (Kamal, 2008). Menurut Zhang dkk. (2008) *embedded system* merupakan sistem yang menggabungkan teknologi komputer, teknologi semikonduktor, teknologi elektronik yang umumnya digunakan pada industri. Sistem tersebut bekerja secara independen dengan mengintegrasikan perangkat lunak dan perangkat keras. Perangkat keras meliputi mikroprosesor, memori, peralatan peripheral, port IO, kontrol grafik, dan lain sebagainya. Perangkat lunak mencakup sistem operasi dan prosedur aplikasi. Prosedur aplikasi digunakan untuk mengontrol kerja sistem, dan sistem operasi digunakan untuk mengontrol prosedur aplikasi dan interaksi perangkat keras. Dalam penelitian ini perangkat keras yang digunakan adalah Arduino.

2.3.1 Arduino

Pengertian Arduino menurut Massimo Banzi (2009), merupakan *platform* dari *physical computing* yang bersifat *open source*. Makna *physical computing* adalah membangun sistem yang dapat menerima rangsangan dari lingkungan dan melakukan respon balik dengan menggunakan perangkat lunak dan perangkat keras.

Arduino merupakan kombinasi dari perangkat keras, bahasa pemrograman dan IDE (*Integrated Development Environment*). Peran dari IDE adalah memungkinkan pengguna arduino untuk menuliskan kode program, yang kemudian akan di-*compile* menjadi kode biner, dan di-*upload* ke dalam memori arduino. Selanjutnya dengan program tersebut, arduino akan berinteraksi dengan dunia luar.

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, karena hardwarenya yang bersifat *open source* sehingga setiap pengguna dapat merangkai arduino sendiri.

2.3.1.1 Perangkat Keras

Berbagai macam bentuk papan arduino telah dirancang hingga saat ini, diantaranya adalah:

- Arduino USB. Menggunakan USB untuk komunikasi dengan computer, contohnya: Arduino Uno, Arduino Duemilanove, Arduino Diecimila, Arduino NG Rev. C, Arduino NG (Nuova Generazione), Arduino Extreme, Arduino Extreme v2, Arduino USB, dan Arduino USB v2.0.
- Arduino Serial. Arduino jenis ini menggunakan RS232 untuk komunikasi dengan komputer. Diantaranya, arduino serial dan arduino serial v2.0.
- Arduino Mega. Papan arduino jenis ini memiliki spesifikasi yang lebih tinggi jika dibandingkan dengan jenis yang lain. Arduino Mega telah dilengkapi dengan tambahan pin digital, pin analog, port serial. Contohnya, Aruduino Mega, Arduino Mega 2560.
- Arduino Fio. Dimanfaatkan untuk penggunaan nirkabel.
- Arduino Lilypad. Merupakan papan arduino dengan bentuk melingkar. Contohnya: Lilypad Arduino 00, Lilypad Arduino 01, Lilypad Arduino 02, Lilypad Arduino 03, Lilypad Arduino 04.
- Arduino BT. Arduino jenis ini memiliki modul Bluetooh yang digunakan untuk komunikasi nirkabel.
- Arduino Nano dan Arduino Mini. Papan arduino jenis ini digunakan bersama breadboard, contohnya: Arduino Nano 3.0, Arduino Nano 2.x, Arduino Mini 04, Arduino Mini 03, dan Arduino Stamp 02.

Komponen utama yang terdapat pada papan arduino adalah Mikrokontroler 8 bit. Jenis mikrokontroler yang digunakan oleh arduino adalah ATmega yang dibuat oleh perusahaan Atmel Corporation. Masing-masing papan arduino memiliki tipe Atmega yang berbeda-beda tergantung dari spesifikasinya, seperti arduino Uno menggunakan ATmega 328, sedangkan arduino Mega 2560 menggunakan ATmega2560.

Penelitian ini menggunakan arduino Mega 2560. Diagram blok arduino Mega 2560 ditunjukkan pada gambar 2.12 yang dijelaskan sebagai berikut:

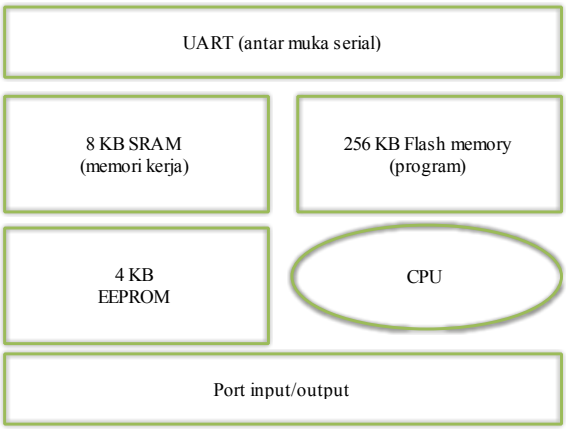
- Universal Asynchronous Receiver/Transmitter (UART) merupakan antar muka yang digunakan untuk komunikasi serial seperti pada RS-232, RS-422, dan RS-485.
- SRAM 8 KB bersifat volatile (hilang saat daya dimatikan), memori ini digunakan oleh variabel-variabel dalam program.
- 256 KB *flash memory* bersifat *non-volatile*, dimanfaatkan untuk menyimpan program yang di-upload dari komputer. *Flash memory* juga menyimpan *bootloader*. *Bootloader* merupakan program inisiasi yang memiliki ukuran kecil, *bootloader* dijalan oleh CPU pada saat daya dihidupkan.
- 4 KB EEPROM bersifat *non-volatile*, digunakan untuk menyimpan data yang tidak boleh hilang saat daya dimatikan.
- *Central Processing Unit* (CPU), merupakan bagian dari mikrokontroler yang digunakan untuk menjalankan setiap instruksi dari program.
- Port input/output, merupakan pin yang digunakan untuk menerima data (*input*) baik untuk pin digital maupun pin analog, dan memberikan data (*output)* baik pada pin digital maupun pin analog.

2.3.1.2 Perangkat Lunak

Komponen lain dari arduino adalah IDE Arduino. IDE Arduino meliputi:

- Editor program, digunakan untuk menulis dan mengedit bahasa Processing. Gambar 2.13 merupakan contoh program pada arduino.

- Compiler, merupakan modul yang dapat mengubah bahasa Processing menjadi kode biner. Karena *microcontroller* hanya dapat memahami kode biner.
- *Uploader*, modul yang digunakan untuk mengunggah kode biner dari komputer ke dalam memori pada papan arduino.



Gambar 2.12 Diagram blok arduino Mega 2560

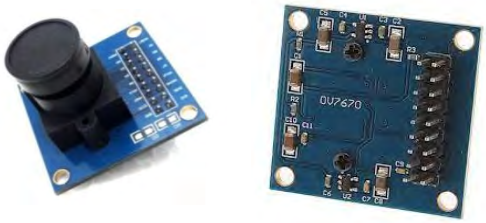
2.3.2 Serial Kamera ov7670 without FIFO

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.

Gambar 2.13 Contoh program arduino

Gambar 2.14 merupakan tampilan depan dan belakang dari serial kamera ov7670 *without FIFO*. Serial kamera ov7670 *without FIFO* memiliki 18 pin. Penjelasan dari masing-masing pin dapat dilihat pada tabel 2.1. Untuk petunjuk penggunaan serial kamera ini dapat dilihat pada "*OV7670/0V7171 CMOS VGA (640X480) CameraChip*^{TM} *Implementation Guide*".



Gambar 2.14 Serial Camera ov7670 *without FIFO*

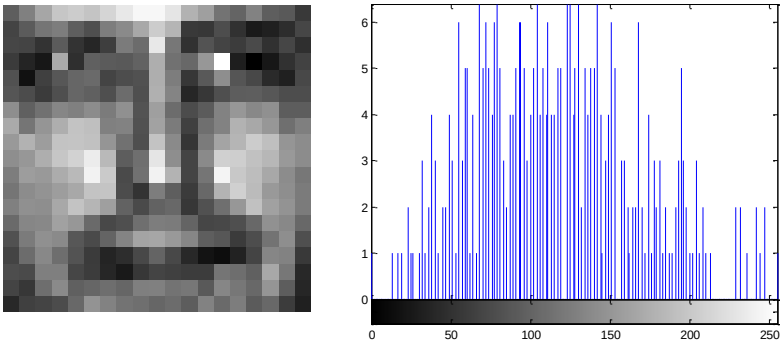
Tabel 2.1 Keterangan pin serial kamera ov7670 *without FIFO*

No.	PIN ov7670	TIPE	KETERANGAN
1.	VDD	*Supply*	*Power supply*
2.	GND	*Supply*	*Ground*
3.	SIOC	Input	SCCB clock
4.	SIOD	Input/output	SCCB data
5.	VSYNC	Output	*Vertical synchronization*
6.	HREF	Output	*Horizontal synchronization*
7.	PCLK	Output	*Pixel clock*
8.	XCLK	Input	*System clock*
9.	D0-D7	Output	*Video parallel output*

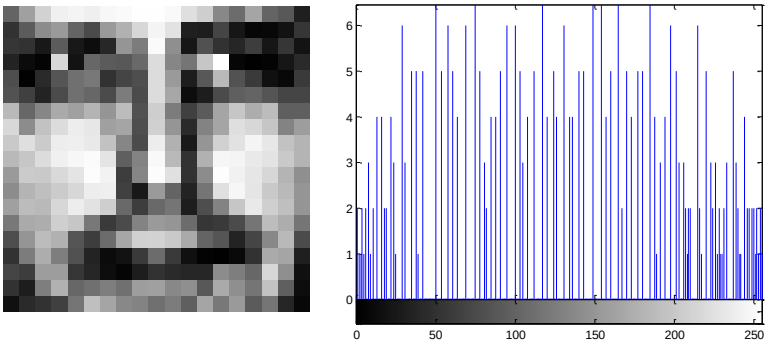
No.	PIN ov7670	TIPE	KETERANGAN
10.	RESET	Input	*Power supply*
11.	PWDN	Input	*Ground*

2.4 Histogram Equalization

Histogram equalization atau perataan histogram merupakan salah satu cara untuk meningkatkan kualitas citra. *Histogram equalization* adalah mengubah nilai-nilai intensitas citra sehingga penyebarannya seragam. Tujuan dari penggunaan *histogram equalization* adalah untuk memperoleh penyebaran histogram yang merata sehingga setiap derajat keabuan memiliki jumlah piksel yang relatif sama. Algoritma dari *histogram equalization* yang digunakan dalam penelitian ini dapat dilihat pada tabel 2.2. Contoh citra hasil dari penerapan algoritma *histogram equalization* dapat dilihat pada gambar 2.16, sedangkan citra aslinya dapat dilihat pada gambar 2.15.



Gambar 2.15 Citra dan histogram sebelum dilakukan *histogram equalization*



Gambar 2.16 Citra dan histrogram setelah dilakukan histogram equalization

Tabel 2.2 Algoritma Histogram Equalization

```
FOR y ← 0 TO L
    r(y) ← y/L
END-FOR
```
Dimana, L adalah derajat keabuan yang terbesar. Sedangkan r adalah normalisasi derajat keabuan.

```
FOR y ← 0 TO N
    FOR x ← 0 TO M
        hist(IMG(y,x)) ← hist(IMG(y,x))+1
    END-FOR
END-FOR
```
Dimana, IMG adalah citra asal, sedangkan N dan M adalah baris dan kolom dari citra asal. Sedangkan untuk hist digunakan untuk menyimpan banyaknya nilai piksel yang muncul dari citra asal.

```
FOR y ← 0 TO L
    prob(y) ← hist(y)/(NxM)
END-FOR
```
Dimana, p adalah probabilitas dari setiap derajat keabuan.

```
cum(0) ← prob(0)
FOR y ← 0 TO L-1
    cum(y+1) ← cum(y-1)+prob(y+1)
END-FOR
```
Dimana, cum adalah frekuensi kumulatif.
```
FOR y ← 0 TO L
    FOR x ← z to L
        histeq(y) ← cum(y) ≈ r(x)
        z ← x
    END-FOR
END-FOR
```
Dimana, histeq digunakan untuk menyimpan nilai piksel yang baru. Nilai piksel tersebut didapatkan dengan cara mencari nilai cum(y) yang mendekati nilai r(x), kemudian menyimpan nilai derajat keabuan dari r(x) pada histeq(y).
```
FOR y ← 0 TO N
    FOR x ← 0 TO M
        output(y,x) ← hist(IMG(y,x))
    END-FOR
END-FOR
```
Dimana, output adalah citra hasil histogram equalization

2.5 Pengertian Algoritma

Pengertian algoritma menurut Shalahuddin dkk. (2007) adalah solusi dari suatu masalah yang harus dipecahkan dengan menggunakan komputer. Algoritma harus dibuat secara runut agar dapat dieksekusi oleh komputer dan memberikan solusi yang diharapkan. Dalam pembuatan sebuah algoritma diperlukan analisis terlebih dahulu dari permasalahan yang akan diselesaikan. Hasil dari analisis

biasanya berupa langkah-langkah apa saja yang perlu dilakukan untuk menyelesaikan sebuah permasalahan.

Secara umum untuk membuat sebuah algoritma diperlukan dua tahap yaitu tahap inisialisasi dan tahap proses. Tahap inisialisasi merupakan tahap yang pertama kali dilakukan sebelum tahap proses. Tahap inisialisasi menjelaskan langkah-langkah awal yang perlu dipersiapkan untuk menyelesaikan permasalahan. Sedangkan tahap proses merupakan urutan dari cara penyelesaian sebuah masalah. Sebagai contoh, bagaimana cara membuat secangkir susu panas, berikut ini adalah langkah-langkah yang harus dilakukan:

- Tahap inisialisasi meliputi:
    1. Menyiapkan cangkir dan sendok
    2. Menyiapkan susu bubuk atau susu kental manis
    3. Menyiapkan gula
    4. Menyiapkan air panas
- Tahap Proses
    5. Memasukkan susu bubuk atau susu kental manis ke dalam cangkir
    6. Memasukkan gula ke dalam cangkir
    7. Memasukkan air panas ke dalam cangkir
    8. Mengaduk susu panas dalam cangkir
    9. Susu panas siap untuk dinikmati.

Langkah-langkah pembuatan secangkir susu panas tersebut tentunya dapat dimengerti oleh manusia. Tetapi langkah-langkah pembuatan secangkir susu panas tersebut tidak dipahami oleh komputer. Komputer hanya memahami bahasa yang dikemas dalam bentuk bahasa pemrograman, seperti JAVA, C, C++, MATLAB, dan lain sebagainya. Untuk memudahkan menerjemahkan langkah-langkah pembuatan secangkir susu panas yang berupa bahasa manusia ke dalam bahasa komputer diperlukan bahasa perantara yaitu bahasa algoritmik (*pseudocode*). Bahasa algoritmik dari langkah-langkah pembuatan secangkir susu panas dapat dilihat pada tabel 2.3.

Tabel 2.3 Langkah-langkah pembuatan secangkir susu dengan bahasa algoritmik

No.	Bahasa Manusia	Bahasa Algoritmik	Keterangan
1	Kondisi awal susu panas belum siap	susuPanas ← 0	Susu panas belum siap.
2	Menyiapkan cangkir dan sendok	cangkir ← 1 sendok ← 1	Cangkir dan sendok yang semula tidak ada menjadi ada, oleh karena itu cangkir diset dengan angka 1.
3	Menyiapkan susu bubuk atau susu kental manis	susu ← 1	Susu yang semula tidak ada menjadi ada, sehingga susu diset 1.
4	Menyiapkan gula	gula ← 1	Gula yang semula tidak ada menjadi ada, sehingga gula diset 1.
5	Menyiapkan air panas	airPanas ← 1	Air panas yang semula tidak ada menjadi ada, sehingga air panas diset 1.
6	Memasukkan susu bubuk atau susu kental manis ke dalam cangkir	cangkir ← cangkir + susu	Susu dimasukkan ke dalam cangkir sehingga dapat dianggap pada cangkir dengan isi keadaan sebelumnya ditambahkan dengan susu.
7	Memasukkan gula ke dalam cangkir	cangkir ← cangkir + gula	Gula dimasukkan ke dalam cangkir sehingga dapat dianggap pada cangkir isi keadaan sebelumnya ditambahkan dengan gula.
8	Memasukkan air panas ke dalam cangkir	cangkir ← cangkir + airPanas	Air panas dimasukkan ke dalam cangkir sehingga dapat dianggap pada cangkir dengan isi keadaan sebelumnya ditambahkan dengan air panas.

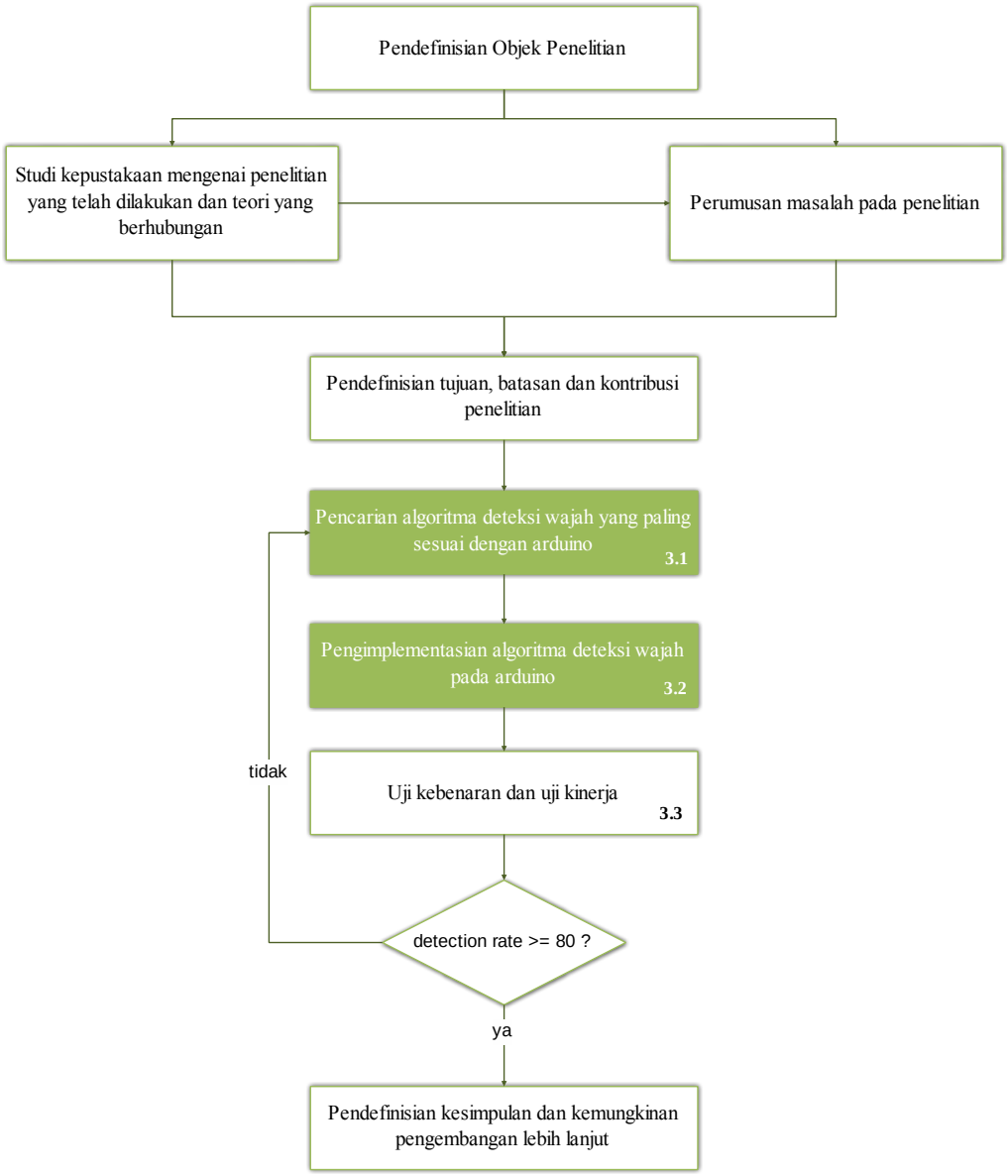
9	Mengaduk susu panas dalam cangkir	While gula = 1 AND susu = 1
		do
		cangkir ← cangkir + sendok
		if cangkir = 50 then
		gula ← 0
		susu ← 0
		end if
		end while
			Selama gula dan susu belum larut maka dilakukan pengadukan, sebagai tanda berhenti jika nilai cangkir telah mencapai 50 setelah terus ditambahkan dengan sendok dalam arti diaduk, maka dianggap gula larut dan diset dengan 0 sehingga proses pengadukan berhenti dan susu panas selesai dibuat.

No.	Bahasa Manusia	Bahasa Algoritmik	Keterangan
10	Susu panas siap untuk dinikmati	susuPanas ← 1	Susu panas siap dinikmati.

Halaman ini sengaja dikosongkan

BAB 3
METODE PENELITIAN

Dalam bab ini dijelaskan mengenai tahapan metode penelitian yang dilakukan. Metode penelitian yang digunakan oleh penulis dapat dilihat pada gambar 3.1. Tahapan dengan warna latar hijau yaitu "Pencarian algoritma deteksi wajah yang paling sesuai dengan arduino" dan "Pengimplementasian algoritma deteksi wajah pada arduino" merupakan kontribusi ilmiah dari penelitian ini.



Gambar 3.1 Metode Penelitian

3.1 Pencarian Algoritma Deteksi Wajah yang Paling Sesuai dengan Arduino

Pencarian algoritma deteksi wajah yang paling sesuai dengan arduino terdiri dari 6 tahap yang diawali dengan tahap memahami kekurangan dan kelebihan dari berbagai tipe arduino. Yang kemudian dilanjutkan dengan memahami kekurangan dan kelebihan dari metode deteksi wajah yang telah ada. Tahap ketiga adalah melakukan pemilihan tipe arduino yang akan digunakan serta memilih metode deteksi wajah yang dapat diimplementasikan pada tipe arduino yang telah dipilih. Tahap ketiga ini dilakukan dengan berdasar pada dua tahap sebelumnya. Tahap selanjutnya adalah memilih kandidat dari algoritma deteksi wajah. Kemudian mengimplementasikan kandidat algoritma deteksi wajah tersebut pada komputer. Untuk menentukan apakah algoritma tersebut layak untuk diimplementasikan pada arduino maka perlu dilakukan langkah keenam yaitu uji kebenaran dan uji kinerja. Jika berdasarkan uji coba tersebut diperoleh *detection rate* lebih besar sama dengan 80%, maka tahap selanjutnya adalah mengimplementasikan algoritma deteksi wajah tersebut pada arduino. Sedangkan jika *detection rate* yang dicapai lebih kecil dari 80% maka kembali pada tahap pemilihan kandidat algoritma deteksi wajah. Untuk lebih jelasnya urutan dalam pencarian algoritma deteksi wajah yang paling sesuai dengan arduino dapat dilihat pada gambar 3.2.

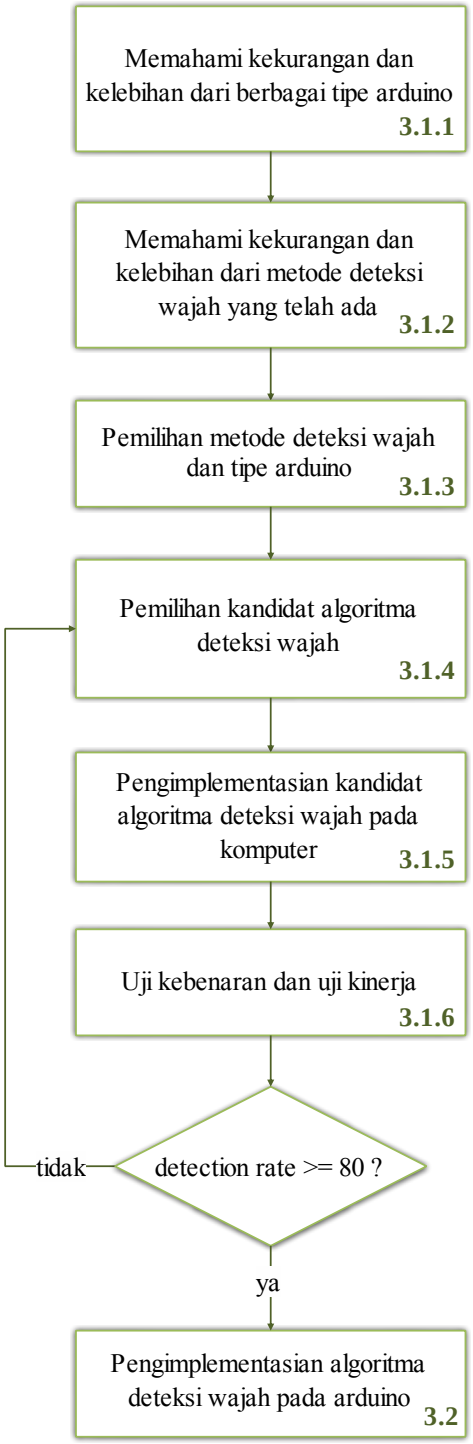
3.1.1 Memahami Kekurangan dan Kelebihan dari Berbagai Tipe Arduino

Hingga saat ini berbagai tipe arduino telah dirilis. Berbagai tipe tersebut dibangun dengan spesifikasi yang berbeda. Sehingga masing-masing tipe arduino tersebut memiliki kelebihan dan kekurangan yang berbeda. Pemilihan dari tipe arduino yang akan digunakan dalam penelitian ini didasarkan pada kelebihan dan kekurangan yang dimiliki oleh masing-masing tipe arduino.

3.1.2 Memahami Kekurangan dan Kelebihan dari Metode Deteksi Wajah yang Telah Ada

Berdasarkan penelitian yang dilakukan oleh Yang dkk. (2002), secara umum algoritma deteksi wajah dikelompokkan dalam empat kategori yaitu metode *knowledge based*, metode *feature invariant*, metode *template matching*,

dan metode *appearance based*. Masing-masing metode tersebut memiliki kelebihan dan kekurangan yang berbeda. Sehingga pemilihan metode yang akan digunakan dalam penelitian ini didasarkan pada kelebihan dan kekurangan tersebut.



Gambar 3.2 Tahapan dalam pencarian algoritma deteksi wajah yang paling sesuai dengan arduino

3.1.3 Pemilihan Metode Deteksi Wajah dan Tipe Arduino

Pemilihan metode deteksi wajah dan tipe arduino dilakukan dengan berdasarkan pada dua sub bab sebelumnya yaitu sub bab memahami kekurangan dan kelebihan dari berbagai tipe arduino dan sub bab memahami kekurangan dan kelebihan dari metode deteksi wajah yang telah ada.

Fokus utama dalam pemilihan metode deteksi wajah selain *detection rate* adalah jumlah memori yang digunakan oleh masing-masing metode tersebut. Sedangkan fokus utama dalam pemilihan tipe arduino adalah jumlah memori yang dimiliki oleh arduino serta kemampuan untuk melakukan komunikasi dengan perangkat keras yang lain (serial kamera ov7670 *without FIFO* dan servo S3003).

3.1.4 Pemilihan Kandidat Algoritma Deteksi Wajah

Kandidat Algoritma deteksi wajah dipilih dengan berdasarkan pada metode deteksi wajah yang telah ditentukan sebelumnya. Penentuan metode deteksi wajah dijelaskan pada sub bab pemilihan metode deteksi wajah dan tipe arduino. Hal utama yang harus diperhatikan dalam pemilihan kandidat algoritma deteksi wajah adalah jenis fitur wajah yang digunakan, algoritma untuk melakukan *training*, dan sistem deteksi wajah yang sederhana.

3.1.5 Pengimplementasian Kandidat Algoritma Deteksi Wajah pada Komputer

Implementasi kandidat algoritma deteksi wajah pada komputer merupakan fase awal yang dilakukan sebelum mengimplementasikan algoritma deteksi wajah pada arduino. Implementasi kandidat algoritma deteksi wajah terdiri dari dua tahap yaitu tahap *training* dan tahap deteksi. Tahap yang dilakukan pertama kali adalah tahap *training*. Pada tahap tersebut dilakukan pelatihan dengan menggunakan citra wajah dan citra bukan wajah. Setelah dilakukan tahap *training,* tahap berikutnya adalah tahap deteksi. Tahap deteksi merupakan tahap yang melakukan identifikasi wajah, dimana identifikasi wajah dilakukan dengan berdasarkan hasil dari tahap *training*. Tahap deteksi wajah tersebut kemudian diuji coba untuk mengetahui kemampuan dari kandidat algoritma deteksi wajah yang diusulkan dalam penelitian ini.

Spesifikasi dari komputer yang digunakan dalam penelitian ini adalah Prosesor Intel Core i5-2450M yang memiliki kecepatan sebesar 2.5 GHz dan dilengkapi dengan memori RAM sebesar 4 GB. Implementasi algoritma deteksi wajah pada komputer dilakukan dengan menggunakan bahasa MATLAB versi 2009a.

3.1.6 Uji Kebenaran dan Uji Kinerja

Uji kebenaran dan uji kinerja pada tahap ini dilakukan untuk menguji algoritma deteksi wajah yang diimplementasikan pada komputer. Tujuan dari uji kebenaran adalah untuk mengetahui *detection rate* dan *false positive* yang dihasilkan oleh algoritma deteksi wajah. Rumus yang digunakan untuk menghitung *detection rate* dapat dilihat pada rumus 2.1 dan untuk menghitung *false positive* ditunjukkan pada rumus 2.2. Sedangkan uji kinerja dilakukan untuk mengetahui kecepatan algoritma deteksi wajah dalam mengidentifikasi wajah. Uji kinerja dihitung dengan menggunakan rumus 2.3.

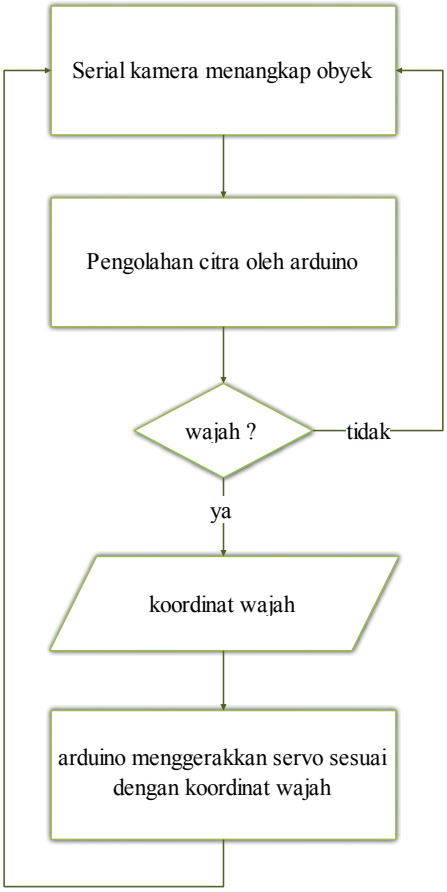
Pada tahap ini target yang ingin dicapai adalah *detection rate* sebesar 80%. Jika target tersebut tercapai, maka akan masuk ke tahap berikutnya yaitu implementasi algoritma deteksi wajah pada arduino. Namun, jika target tersebut tidak tercapai langkah yang dilakukan adalah kembali pada tahap pemilihan kandidat algoritma deteksi wajah.

3.2 Pengimplementasian Algoritma Deteksi Wajah pada Arduino

Implementasi algoritma deteksi wajah pada arduino dilakukan dengan menggunakan tiga perangkat keras yaitu serial kamera ov7670 *without FIFO*, arduino, dan servo S3003. Serial kamera digunakan untuk menangkap obyek. Arduino digunakan untuk menjalankan algoritma deteksi wajah serta komunikasi dengan serial kamera dan servo. Sedangkan servo digunakan untuk menggerakkan kamera secara horisontal dan vertikal.

Urutan proses deteksi wajah dengan menggunakan arduino dapat dilihat pada gambar 3.3. Langkah awal dari proses ini adalah menangkap obyek dengan menggunakan serial kamera, kemudian diproses dengan menggunakan arduino sehingga menjadi citra digital. Selanjutnya, Arduino mengolah citra tersebut

dengan menggunakan algoritma deteksi wajah yang telah dipilih sebelumnya. Jika pada citra tersebut terdapat wajah maka algoritma deteksi wajah tersebut akan mengeluarkan koordinat wajah yang terdapat pada citra. Dan arduino akan menggerakkan servo secara horisontal dan vertikal sesuai dengan koordinat yang telah diperoleh. Tujuan dari penggunaan servo adalah untuk mengejar posisi wajah, sehingga serial kamera dapat mengikuti gerakan wajah. Namun apabila pada citra tersebut tidak terdapat wajah, maka kamera akan kembali mengambil obyek. Urutan proses ini akan selalu berulang selama arduino mendapatkan arus listrik.



Gambar 3.3 Urutan proses deteksi wajah dengan menggunakan arduino

3.3 Uji Kebenaran dan Uji Kinerja

Pada tahap ini uji kebenaran dan uji kinerja dilakukan untuk menguji algoritma deteksi wajah pada saat diimplementasikan dengan menggunakan arduino. Uji kebenaran dilakukan untuk mengetahui *detection rate* dan *false*

positive dari algoritma deteksi wajah pada saat diimplementasikan dengan menggunakan arduino. Rumus yang digunakan untuk menghitung *detection rate* dapat dilihat pada rumus 2.1 dan untuk menghitung *false positive* ditunjukkan oleh rumus 2.2. Sedangkan uji kinerja dilakukan untuk mengetahui kecepatan dari arduino dalam melakukan identifikasi wajah, dimana kecepatan dihitung mulai dari pengambilan obyek oleh serial kamera hingga algoritma deteksi wajah melakukan identifikasi wajah. Uji kinerja dihitung dengan menggunakan rumus 2.3.

Target yang ingin dicapai dari penelitian ini adalah arduino mampu mengidentifikasi wajah dengan nilai *detection rate* sebesar 80%. Jika target tersebut tidak tercapai langkah yang akan dilakukan adalah kembali pada tahap pencarian algoritma deteksi wajah yang paling sesuai dengan arduino.

Halaman ini sengaja dikosongkan

BAB 4
HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini dijelaskan mengenai langkah-langkah yang dilakukan untuk mencari algoritma deteksi wajah yang paling sesuai dengan arduino, implementasi algoritma deteksi wajah pada arduino, dan uji kebenaran serta uji kinerja.

4.1 Pencarian Algoritma Deteksi Wajah yang Paling Sesuai dengan Arduino

Seperti yang telah ditunjukkan pada gambar 3.2 ada 6 tahap yang harus dilakukan untuk mendapatkan algoritma deteksi wajah yang paling sesuai dengan arduino. Dua tahap awal merupakan tahap terpenting yang harus dilakukan terlebih dahulu sebelum membangun sistem deteksi wajah dengan menggunakan arduino. Tahap yang pertama adalah memahami kekurangan dan kelebihan dari berbagai tipe arduino, sedangkan tahap yang kedua adalah memahami kekurangan dan kelebihan dari metode deteksi wajah yang telah ada. Setelah dua tahap tersebut dilakukan, langkah berikutnya adalah memilih metode deteksi wajah dan tipe arduino. Kemudian tahap selanjutnya adalah memilih kandidat algoritma deteksi wajah. Kandidat algoritma deteksi wajah tersebut kemudian diimplementasikan dengan menggunakan komputer. Tahap implementasi kandidat algoritma deteksi wajah terdiri dari dua bagian yaitu *training* dan deteksi. Tahap berikutnya setelah mengimplementasikan kandidat algoritma deteksi wajah adalah melakukan uji coba pada kandidat algoritma deteksi wajah tersebut. Uji coba meliputi uji kebenaran dan uji kinerja

4.1.1 Memahami Kekurangan dan Kelebihan dari Berbagai Tipe Arduino

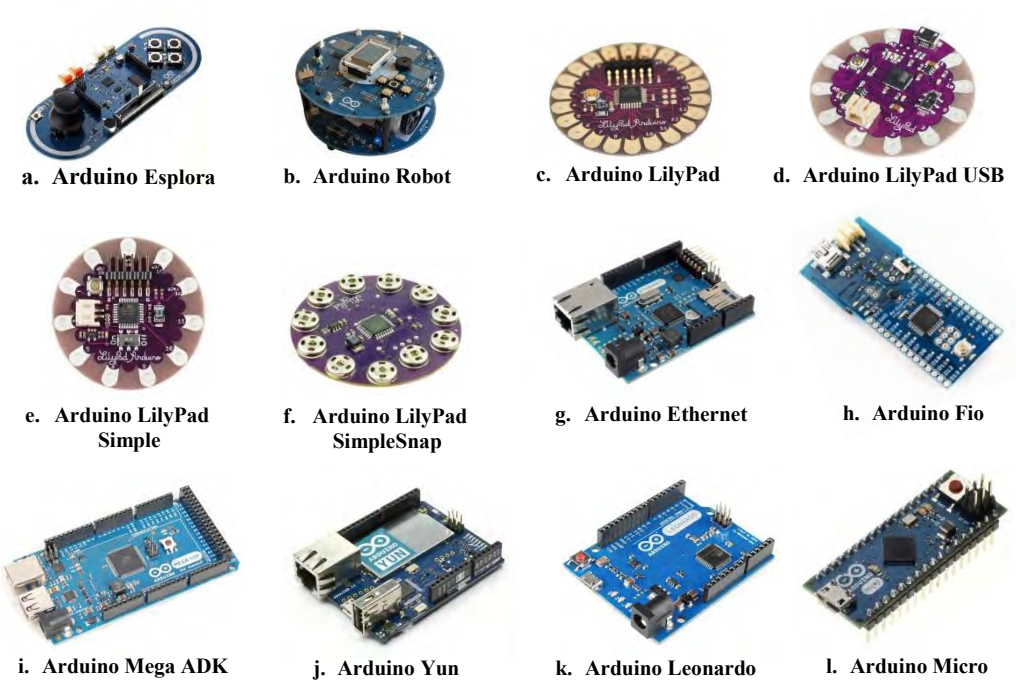
Hingga saat ini berbagai tipe arduino telah di buat. Masing-masing tipe arduino didesain untuk kebutuhan yang berbeda, ada sebagian tipe arduino yang didesain untuk keperluan yang spesifik dan ada juga yang didesain untuk keperluan yang umum. Beberapa tipe arduino seperti arduino Esplora, arduino Robot, arduino LilyPad, arduino LilyPad USB, arduino LilyPad Simple, arduino LilyPad SimpleSnap, arduino Ethernet, arduino Fio, arduino ADK, arduino Yun, arduino Leonardo, dan arduino Micro merupakan arduino yang didesain untuk

keperluan khusus. Arduino Esplora (gambar 4.1 bagian a) dilengkapi dengan pengontrol sehingga lebih cocok digunakan untuk sistem yang memerlukan penggerak manual. Arduino Robot (gambar 4.1 bagian b) didesain dengan motor sehingga lebih cocok untuk mobil cerdas. Arduino LilyPad (gambar 4.1 bagian c), arduino LilyPad USB (gambar 4.1 bagian d), arduino LilyPad Simple (gambar 4.1 bagian e), arduino LilyPad SimpleSnap (gambar 4.1 bagian f) didesain untuk keperluan industri tekstil. Arduino Ethernet (gambar 4.1 bagian g) didesain untuk sistem yang memerlukan modul ethernet. Arduino Fio (gambar 4.1 bagian h) lebih cocok digunakan untuk sistem yang memerlukan penggunaan *wireless*. Arduino ADK (gambar 4.1 bagian i) didesain untuk berkomunikasi dengan android. Arduino Yun (gambar 4.1 bagian j) didesain untuk sistem yang memerlukan modul jaringan. Arduino Leonardo (gambar 4.1 bagian k) dan arduino Micro (gambar 4.1 bagian l) lebih cocok digunakan untuk sistem yang berkomunikasi dengan komputer. Dalam penelitian ini digunakan tipe arduino yang didesain untuk kepentingan umum, seperti arduino Diecimila (gambar 4.2 bagian a), arduino Mini (gambar 4.2 bagian b), arduino Nano (gambar 4.2 bagian c), arduino Pro (gambar 4.2 bagian d), arduino Pro Mini (gambar 4.2 bagian e), arduino Duemilanove (gambar 4.2 bagian f), arduino Uno (gambar 4.2 bagian g), arduino Mega 1280 (gambar 4.2 bagian h), arduino Mega 2560 (gambar 4.2 bagian i), dan arduino Due (gambar 4.2 bagian j). Namun tipe arduino tersebut memiliki prosesor, kecepatan, jumlah pin, flash memory, SRAM, dan EEPROM yang berbeda. Perbedaan tersebut dapat dilihat pada tabel 4.1. Dari tabel tersebut dapat diketahui kelebihan dan kekurangan dari masing-masing tipe arduino.

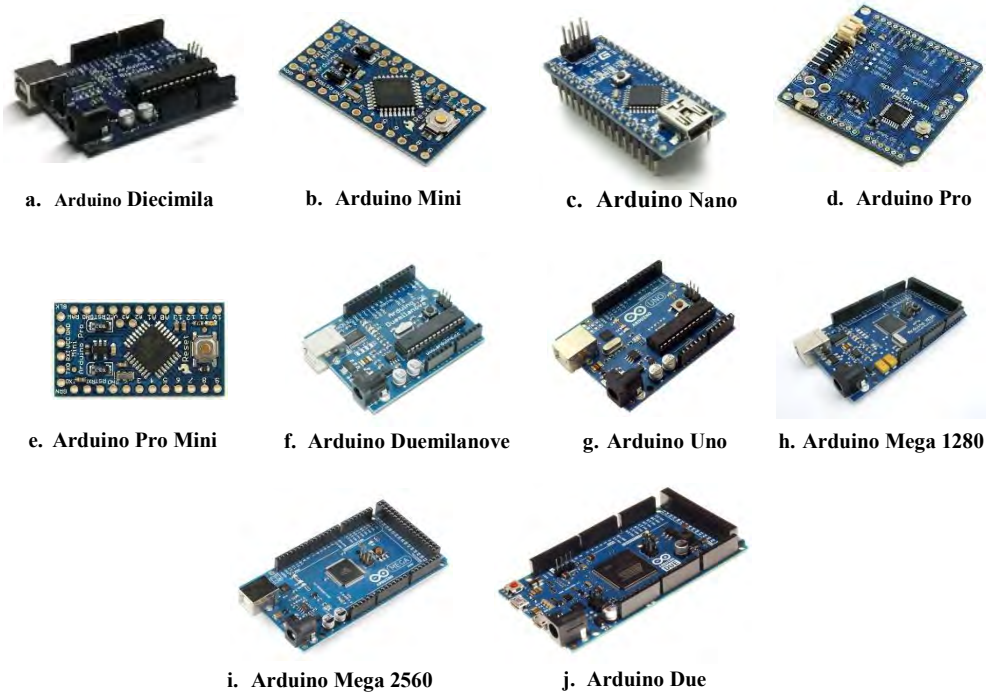
Tabel 4.1 Perbandingan tipe Arduino

No.	Tipe Arduino Prosesor	Kec.	Pin Digital I/O / PWM	Pin Analog	*Flash Memory* (KB)	SRAM (KB)	EEPROM
1	**Pro 168** ATmega168	8 (3.3V) 16 (5V)	14 / 6	6	16	1	512 bytes
2	**Diecimila** ATmega168	16	14 / 6	6	16	1	512 bytes
3	**Nano 168** ATmega168	16	14 / 6	8	16	1	512 bytes
4	**Duemilanove 168** ATmega168	16	14 / 6	6	16	1	512 bytes

No.	Tipe Arduino Prosesor	Kec.	Pin Digital I/O / PWM	Pin Analog	*Flash Memory* (KB)	SRAM (KB)	EEPROM
5	**Pro Mini** ATmega168	8 (3.3V) 16 (5V)	14 / 6	8	16	1	512 bytes
6	**Nano 328** ATmega328	16	14 / 6	8	32	2	1 KB
7	**Pro 328** ATmega328	8 (3.3V) 16 (5V)	14 / 6	6	32	2	1 KB
8	**Mini** ATmega328	16	14 / 6	8	32	2	1 KB
9	**Duemilanove 328** ATmega328	16	14 / 6	6	32	2	1 KB
10	**Uno** ATmega328	16	14 / 6	6	32	2	1 KB
11	**Mega 1280** ATmega1280	16	54 / 15	16	128	8	4 KB
12	**Mega 2560** ATmega2560	16	54 / 15	16	256	8	4 KB
13	**Due** AT91SAM3X8E	84	54 / 12	12	512	96	-



Gambar 4.1 Tipe arduino yang didesain untuk keperluan tertentu



Gambar 4.2 Tipe arduino yang didesain untuk keperluan umum

4.1.2 Memahami Kekurangan dan Kelebihan dari Metode Deteksi Wajah yang Telah Ada

Metode deteksi wajah terbagi menjadi empat kelompok yaitu *knowledge based, feature invariant*, *template matching*, dan *appearance based* (Yang dkk., 2002). Berikut ini merupakan penjelasan singkat dari masing-masing metode tersebut:

- **Metode Knowledge Based**

  Metode *knowledge based* merupakan metode dengan pendekatan *top-down*. Dimana, fitur yang digunakan adalah fitur sederhana yang terdapat pada wajah manusia seperti keberadaan dua mata yang simetris, hidung dan mulut. Proses identifikasi wajah dilakukan secara bertingkat yang secara umum terbagi dalam dua tahap yaitu pencarian kandidat wajah dan verifikasi. Pada tahap yang pertama, kandidat wajah diperoleh dengan menerapkan aturan tertentu terhadap fitur wajah. Aturan yang digunakan dapat berupa jarak antar fitur atau posisi dari masing-masing fitur. Tahap berikutnya setelah diperoleh kandidat wajah dilakukan verifikasi untuk

mengurangi *false positive*. Salah satu penelitian yang menggunakan metode ini adalah penelitian yang dilakukan oleh Yang dan Huang (1994).

- **Metode Feature Invariant**

  Metode *feature invariant* merupakan kebalikan dari metode *knowledege based*. Metode ini menggunakan pendekatan *bottom-up.* Disebut *bottom-up* karena diawali dengan pencarian fitur yang dapat digunakan untuk mendeteksi wajah. Fitur yang digunakan adalah fitur yang dapat mengidentifikasi wajah dengan berbagai posisi kemiringan, berbagai warna latar belakang yang berbeda, dan berbagai kondisi pencahayaan. Secara umum proses identifikasi wajah dilakukan dengan cara menggunakan fitur wajah dan kemudian memutuskan apakah ada wajah atau tidak. Fitur wajah yang digunakan adalah alis mata, mata, hidung, mulut, dan rambut. Beberapa penelitian yang menggunakan metode ini adalah Leung dkk. (1995), Yow dan Cipolla (1997), Dai dan Nakano (1996), Yang dan Waibel (1996), McKenna dkk. (1998), dan Kjeldsen dan Kender (1996).

- **Metode Template Matching**

  Pada metode *template matching* proses identifikasi wajah dilakukan dengan menggunakan *face pattern*. Umumnya *face pattern* dapat berupa mata, hidung, dan mulut yang didefinisikan secara terpisah. Keberadaan wajah dari sebuah citra ditentukan oleh nilai dari setiap *face pattern* yang digunakan. Penelitian yang menggunakan *template matching* adalah Craw dkk. (1992) dan Lanitis dkk. (1995).

- **Metode Appearance Based**

  Berbeda dengan metode *template matching*, metode *appearance based* menggunakan *face pattern* yang didapatkan dari hasil *training* dengan menggunakan citra wajah dan citra bukan wajah. Beberapa penelitian yang menggunakan metode *appearance based* adalah Turk dan Pentland (1991), Sung dan Poggio (1998), Rowley dkk. (1998), Osuna dkk. (1997), Schneiderman dan Kanade (1998), Rajagopalan dkk. (1998), Lew (1996), Colmenarez dan Huang (1997), dan Viola dan Jones (2004).

Kekurangan dan kelebihan dari masing-masing metode deteksi wajah tersebut dijelaskan pada tabel 4.2.

Tabel 4.2 Kekurangan dan kelebihan 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.

4.1.3 Pemilihan Metode Deteksi Wajah dan Tipe Arduino

Pemilihan metode deteksi wajah dan tipe arduino dilakukan dengan cara memahami kekurangan dan kelebihan dari metode deteksi wajah serta memahami kekurangan dan kelebihan dari tipe arduino. Kekurangan dan kelebihan dari metode deteksi wajah telah dijelaskan pada sub bab 4.1.2, sedangkan kekurangan dan kelebihan dari tipe arduino dijelaskan pada sub bab 4.1.1. Kriteria dalam pemilihan metode deteksi wajah sebagai berikut:

1. Mudah diimplementasikan pada arduino.
2. Proses identifikasi wajah memerlukan memori dalam jumlah kecil.

Sedangkan kriteria dalam pemilihan tipe arduino sebagai berikut:

1. Dapat berinteraksi dengan serial kamera ov7670 *without FIFO* dan servo S3003.
2. Memiliki memori yang cukup untuk melakukan identifikasi wajah.

Pada tabel 4.1 dijelaskan mengenai spesifikasi dari tipe arduino. Susunan pada tabel 4.1 dibuat secara berurutan dari spesifikasi terendah ke spesifikasi tertinggi. Berdasarkan tabel tersebut dapat dilihat bahwa tipe arduino yang memiliki kecepatan paling tinggi dan memori paling banyak adalah arduino due, dimana arduino due memiliki kecepatan sebesar 84 MHz, *flash memory* sebesar 512 KB, dan SRAM sebesar 86 KB. Namun, permasalahannya adalah saat penelitian ini dilakukan modul I^2C yang digunakan untuk berinteraksi dengan serial kamera ov7670 *without FIFO* belum dapat digunakan, sehingga pilihan jatuh pada peringkat berikutnya yaitu arduino Mega 2560. Dimana arduino Mega 2560 dapat berinteraksi dengan servo S3003 dan telah dilengkapi dengan modul I^2C sehingga dapat berkomunikasi dengan serial kamera ov7670 *without FIFO* (spesifikasi serial kamera ov7670 *without FIFO* dapat dilihat pada sub bab 2.3.2). Sedangkan untuk arduino versi dibawahnya tidak cocok untuk digunakan dalam penelitian ini karena jumlah pin yang dimiliki tidak memadai untuk berkomunikasi dengan serial kamera ov7670 *without FIFO* dan jumlah memori yang dimiliki kurang banyak (karena untuk menyimpan citra hasil dari serial kamera ov7670 *without FIFO* diperlukan memori sebesar 1200 bytes, sehingga tentunya akan diperlukan memori lebih dari 1200 bytes agar sistem deteksi wajah dapat berjalan).

Berdasarkan tabel 4.2, metode *knowledge based* dan *feature invariant* tidak cocok digunakan dalam penelitian ini, karena metode tersebut hanya dapat melakukan identifikasi dengan baik jika dalam sebuah citra hanya terdapat satu wajah dan jika lebih dari satu wajah maka metode tersebut tidak berjalan dengan optimal. Sehingga metode yang memungkinkan untuk diimplementasikan pada arduino adalah metode *template matching* dan metode *appearance based*.

Sistem deteksi wajah dengan menggunakan metode *template matching* yang diimplementasikan pada *embedded system* telah dilakukan oleh Hori dkk. (2004). Pada penelitian tersebut perangkat keras yang digunakan adalah FPGA dan mikrokontroler HITACHI SH-3. Konsep dari penelitian tersebut dapat dilihat pada sub bab 2.2.2. Konsep tersebut terlalu rumit untuk diimplementasikan pada arduino, dikarenakan tipe arduino yang ada saat ini sangat terbatas dalam jumlah memori dan kecepatan (tabel 4.1).

Sedangkan implementasi sistem deteksi wajah dengan metode *appearance based* pada *embedded system* telah dilakukan oleh Kyrkou dkk. (2010) dan He dkk. (2009). Perangkat keras yang digunakan oleh Kyrkou dkk (2010) adalah Xilink Virtex II Pro FPGA, sedangkan yang digunakan oleh He dkk. (2009) adalah Xilinx FX 130T Virtex5 FPGA. Konsep dari penelitian yang dilakukan oleh Kyrkou dkk. (2010) dan He dkk. (2009) dijelaskan pada sub bab 2.2.3. Konsep dari kedua penelitian tersebut mengacu pada konsep yang diusulkan oleh Viola dan Jones (2004). Secara umum proses identifikasi wajah yang dijelaskan pada konsep tersebut dilakukan secara bertahap dan menggunakan fitur berupa *haar feature*. Penggunaan *haar feature* mengharuskan penyimpanan fitur dalam jumlah yang sangat banyak sehingga untuk mengimplementasikan konsep tersebut diperlukan perangkat keras yang memiliki memori yang cukup besar dan mampu menjalankan komputasi dengan cepat.

Sehingga tampak bahwa metode *appearance based* memerlukan memori yang besar dan tidak cocok untuk digunakan dalam penelitian ini, karena SRAM yang dimiliki oleh arduino Mega 2560 hanya sebesar 8 KB. Oleh karena itu metode yang mungkin digunakan adalah metode *template matching*. Metode *template matching* yang dilakukan oleh Hori dkk. (2004) terdiri dari tiga tahap

yaitu sebelum pemrosesan citra, mendeteksi wajah dengan menggunakan SSGA, dan menetapkan wajah dengan mencari daerah mulut. Pada tahap sebelum pemrosesan citra dilakukan ekstrasi fitur dan deteksi *edge*, kemudian dilanjutkan dengan tahap berikutnya yaitu mencari kandidat wajah pada citra dengan menggunakan SSGA, dan selanjutnya melakukan verifikasi keberadaan wajah dengan cara mencari keberadaan mulut pada kandidat wajah. Tiga tahap tersebut cukup rumit jika diimplementasikan pada arduino mega 2560, karena arduino mega 2560 memiliki kecepatan dan memori yang terbatas. Oleh karena itu pada penelitian ini diusulkan metode *template matching* menggunakan fitur berupa *template* wajah yang berbentuk seperti topeng, fitur tersebut dapat dilihat pada gambar 4.3.

Gambar 4.3 *Template* wajah yang digunakan sebagai fitur

4.1.4 Pemilihan Kandidat Algoritma Deteksi Wajah

Kandidat 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 4.3. 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.

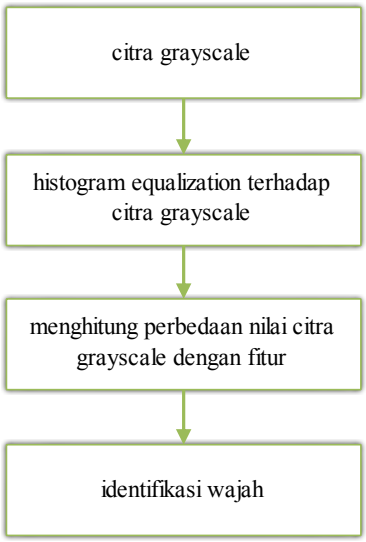
Pada umumnya kondisi obyek yang ditangkap oleh kamera dipengaruhi oleh berbagai kondisi pencahayaan yang berbeda. Kondisi cahaya yang terlalu terang mengakibatkan citra berwarna lebih terang, sedangkan kondisi cahaya yang terlalu gelap mengakibatkan citra berwarna lebih gelap. Serta posisi datangnya 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*. Konsep *histogram equalization* dijelaskan pada sub bab 2.4.

Secara runut kandidat algoritma deteksi wajah yang diusulkan dalam penelitian ini dijelaskan pada gambar 4.4. 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 kandidat algoritma deteksi wajah meliputi dua tahap yaitu tahap *training* dan tahap deteksi, kedua tahap tersebut dijelaskan pada sub bab 4.1.5.

4.1.5 Pengimplementasian Kandidat Algoritma Deteksi Wajah pada Komputer

Pengimplementasian kandidat algoritma deteksi wajah pada komputer diawali dengan tahap *training* kemudian dilanjutkan dengan tahap deteksi. Tujuan dari tahap *training* adalah untuk mendapatkan fitur (berupa *template* wajah) yang memiliki kemampuan identifikasi wajah yang baik. Setelah diperoleh fitur, langkah selanjutnya adalah melakukan tahap deteksi wajah. Pada tahap deteksi wajah dilakukan proses identifikasi wajah dengan menggunakan fitur yang telah didapatkan pada tahap *training*. Tahap deteksi wajah tersebut kemudian diuji coba

untuk mengetahui kemampuan dari algoritma deteksi wajah yang diusulkan dalam penelitian ini.



Gambar 4.4 Konsep kandidat algoritma deteksi wajah

4.1.5.1 Tahap 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 4.5, sedangkan contoh citra bukan wajah dapat dilihat pada gambar 4.6. Secara umum langkah-langkah yang dilakukan pada tahap *training* dapat dilihat pada gambar 4.7 yang dijelaskan sebagai berikut:

1. **Mencari posisi template**

   Tahap ini bertujuan untuk mencari posisi baris dan kolom yang optimal dari *template* wajah pada citra wajah. *Template* wajah yang digunakan pada tahap ini berbentuk segi empat dengan ukuran a x b, bentuk setiap *template* wajah dapat dilihat pada gambar 4.8, sedangkan ukuran setiap *template* wajah dapat dilihat pada tabel 4.3. Jumlah template wajah yang di-*training* pada penelitian ini adalah 39 *template* dengan ukuran maksimal sebesar 19x19 piksel.

Untuk mendapatkan posisi baris dan kolom yang optimal, setiap *template* wajah di-*training* dengan 2000 citra wajah. Tahap ini dilakukan per *template* wajah. Langkah-langkah yang dilakukan untuk mendapatkan posisi baris dan kolom dapat dilihat pada lampiran A tahap 1 yang dijelaskan sebagai berikut:

a. Membuat array untuk menyimpan total nilai perbedaan dari citra wajah dengan *template* wajah, dan setiap posisi baris dan kolom dari array tersebut menggambarkan posisi *template* wajah pada citra. Ukuran array untuk setiap *template* wajah berbeda-beda, banyaknya baris dihitung dengan menggunakan rumus 4.1, sedangkan banyaknya kolom dihitung dengan menggunakan rumus 4.2.

$$baris\ array = 19 - baris\ template\ wajah + 1 \quad (4.1)$$

$$kolom\ array = 19 - kolom\ template\ wajah + 1 \quad (4.2)$$

b. Melakukan *histogram equalization* pada citra wajah dengan menggunakan algoritma yang terdapat pada tabel 2.2.

c. Perhitungan posisi *template* wajah dimulai dari koordinat (1,1) (koordinat array pada matlab dimulai dari (1,1)), posisi 1 yang pertama adalah baris, sedangkan posisi 1 yang kedua adalah kolom. Selanjutnya dilakukan pemotongan citra sesuai dengan panjang baris dan kolom dari *template* wajah yang digunakan. Kemudian dihitung nilai perbedaan antara *template* wajah dengan potongan citra tersebut, nilai perbedaan tersebut disimpan dengan cara dijumlahkan pada array yang telah dibuat sebelumnya di koordinat (1,1). Setelah nilai perbedaan tersebut disimpan, perhitungan posisi *template* wajah dilanjutkan pada kolom berikutnya hingga batas akhir dari kolom, posisi batas akhir dari kolom sama dengan banyaknya kolom array yang dihitung dengan menggunakan rumus 4.2, jika telah berada pada posisi batas akhir dari kolom maka dilanjutkan pada baris berikutnya yang dimulai dari posisi kolom 1 hingga mencapai posisi maksimal dari kolom, kemudian bergerak ke baris berikutnya lagi hingga mencapai posisi akhir dari baris dan posisi akhir dari kolom, posisi akhir dari baris sama dengan banyaknya baris pada array yang dihitung dengan menggunakan rumus 4.1.

d. Mengulangi kembali langkah b dengan menggunakan citra wajah berikutnya, hal ini dilakukan berulang kali hingga 2000 citra wajah.

e. Mencari total nilai yang paling rendah dari array. Dimana koordinat dari total nilai yang paling rendah adalah posisi baris dan kolom yang optimal dari *template* wajah pada citra.

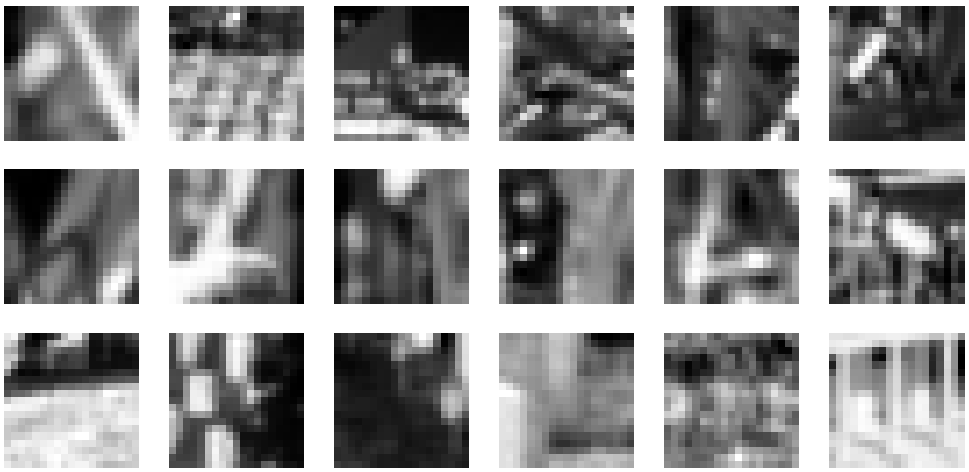
Gambar 4.5 Citra wajah

2. Menghitung nilai *threshold* awal

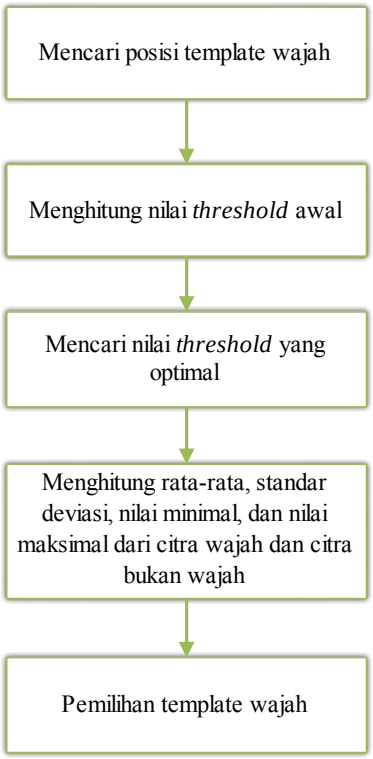
Setelah mendapatkan koordinat baris dan kolom dari *template* wajah, langkah selanjutnya adalah menghitung nilai *threshold* awal. Nilai *threshold* tersebut akan digunakan pada tahap berikutnya yaitu mencari nilai *threshold* yang optimal. Proses pencarian nilai *threshold* awal dilakukan per *template* wajah dengan menggunakan seluruh citra wajah dan citra bukan wajah. Algoritma yang digunakan untuk menghitung nilai *threshold* awal dapat dilihat pada lampiran A tahap 2 yang dijelaskan sebagai berikut:

a. Menghitung rata-rata dari nilai perbedaan antara *template* wajah dengan setiap 2000 citra wajah. Posisi dari *template* wajah pada citra wajah sesuai dengan posisi baris dan kolom yang telah didapatkan pada tahap 1.
b. Menghitung rata-rata dari nilai perbedaan antara *template* wajah dengan setiap 4000 citra bukan wajah. Posisi dari *template* wajah pada citra bukan wajah sesuai dengan posisi baris dan kolom yang telah didapatkan pada tahap 1.
c. Nilai *threshold* awal didapatkan dengan cara menjumlahkan nilai dari langkah a dan b kemudian dibagi 2 (rumus 4.3).

$$threshold\ awal = \frac{mean\ citra\ wajah + mean\ citra\ bukan\ wajah}{2} \qquad (4.3)$$



Gambar 4.6 Citra bukan wajah



Gambar 4.7 Langkah-langkah untuk melakukan *training*

3. 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 *thershold* 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 lampiran A tahap 3 yang dijelaskan sebagai berikut:

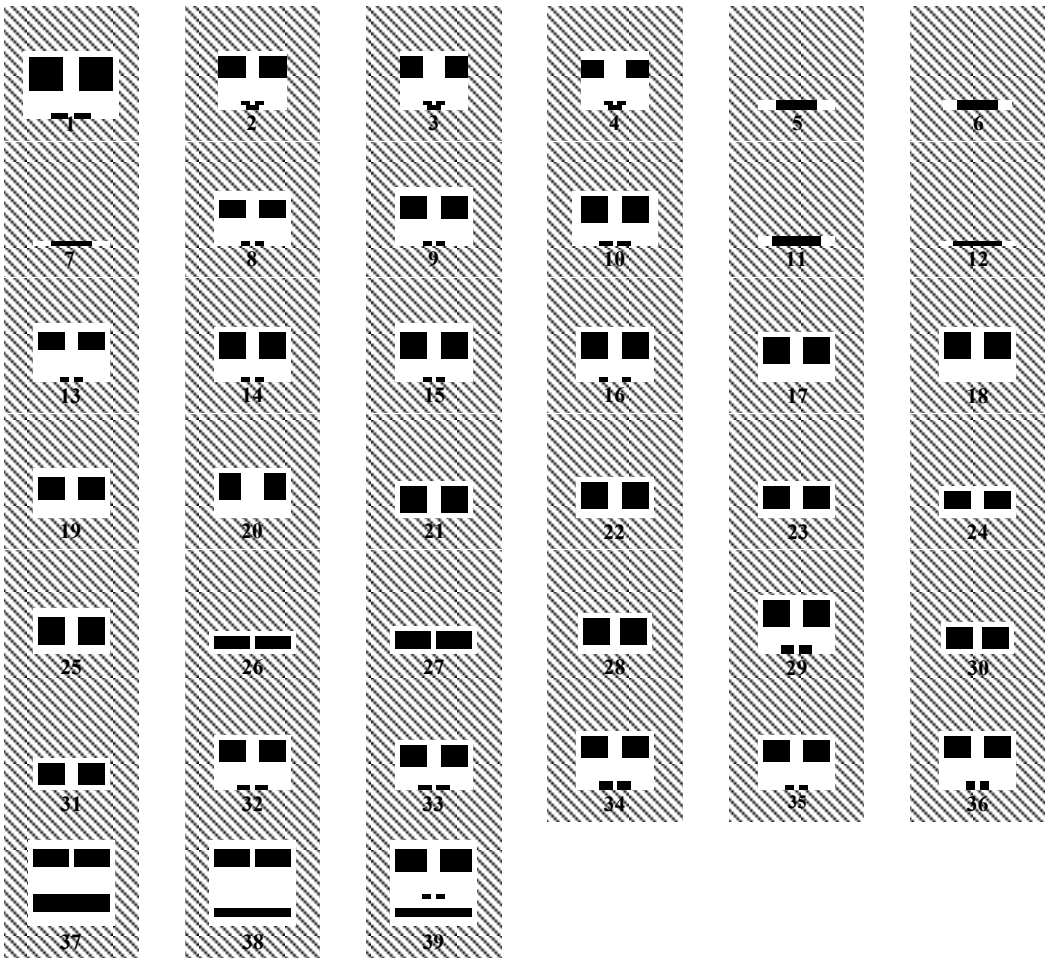
a. Membandingkan nilai perbedaan antara *template* wajah dan citra wajah dengan *threshold*, jika nilai *threshold* lebih kecil dari nilai perbedaan antara *template* wajah dan citra berarti keberadaan wajah pada citra wajah tersebut tidak teridentifikasi, maka nilai *true negative* (TN) ditambah 1, langkah ini dilakukan hingga 2000 citra wajah.
b. Membandingkan nilai perbedaan antara *template* wajah dan citra bukan wajah dengan *threshold*, jika nilai *threshold* lebih besar dari nilai perbedaan antara *template* wajah dan citra berarti teridentifikasi keberadaan wajah pada citra bukan wajah, maka nilai *false positive* (FP) ditambah 1, langkah ini dilakukan hingga 4000 citra bukan wajah.
c. Menghitung total kesalahan (sumErr) dengan menggunakan rumus 4.4.

$$total\ kesalahan = \frac{(false\ positive + true\ negative)}{total\ citra\ wajah + total\ citra\ bukan\ wajah} \quad (4.4)$$

d. Melakukan perbandingan antara nilai sumErr sebelumnya dengan nilai sumErr terbaru, jika tidak ada perbedaan pada nilai sumErr tersebut selama 20 iterasi maka telah didapatkan nilai *threshold* yang optimal sehingga langkah selanjutnya adalah mencari *threshold* yang optimal untuk *template* wajah berikutnya dan proses ini terus berulang hingga mencapai 39 *template* wajah, namun jika ada perbedaan pada nilai sumErr tersebut baik lebih kecil maupun lebih besar maka akan melanjutkan pada langkah e.
e. Membandingkan nilai *false positive* dengan nilai *true negative.* Nilai *false positive* dihitung dengan menggunakan rumus 4.5 sedangkan nilai *true negatif* dihitung dengan menggunakan rumus 4.6. Jika nilai *true negative* lebih besar dari nilai *false positive* maka nilai *threshold* ditambah 1, jika nilai *true negative* lebih kecil dari nilai *false positive* maka nilai *threshold* di kurangi 1. Setelah mendapatkan nilai *threshold* yang baru maka akan kembali pada langkah a.

$$false\ positive = \frac{total\ false\ positive}{jumlah\ citra\ bukan\ wajah} \quad (4.5)$$

$$true\ negative = \frac{total\ true\ negatif}{jumlah\ citra\ wajah} \quad (4.6)$$



Gambar 4.8 *Template* wajah

Tabel 4.3 Ukuran *template* wajah

Nomor Fitur	Baris	Kolom
1	12	17
2	13	15
3	13	15
4	13	15
5	2	17
6	2	15
7	1	17
8	12	17
9	13	17
10	12	19

Nomor Fitur	Baris	Kolom
11	2	17
12	1	17
13	13	17
14	12	17
15	13	17
16	12	17
17	11	17
18	12	17
19	11	17
20	11	17
21	8	17
22	9	17
23	8	17
24	7	17
25	10	17
26	5	19
27	6	19
28	9	16
29	13	17
30	7	16
31	7	17
32	12	17
33	11	17
34	13	17
35	12	17
36	13	17
37	19	19
38	19	19
39	19	19

4. 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. Proses untuk mendapatkan nilai *threshold* yang akan digunakan untuk mendeteksi wajah dilakukan secara manual yang dijelaskan pada sub bab 4.1.6.2.1.

5. 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 4.6, sedangkan *false positive* dihitung dengan menggunakan rumus 4.5. 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.

Sebagai contoh dari penerapan langkah-langkah pada gambar 4.7, dilakukan implementasi dari langkah-langkah tersebut dengan menggunakan *template* wajah nomor 9 yang dapat dilihat pada gambar 4.8 dan citra wajah yang ditunjukkan pada gambar 4.10. Ukuran citra wajah adalah 19x19 piksel, sedangkan ukuran dari *template* wajah adalah 13x17 piksel. Kemudian dilakukan *histogram equalization* terhadap citra wajah, hasil dari *histogram equalization* dapat dilihat pada gambar 4.11. Selanjutnya menghitung perbedaan nilai *template* wajah dengan citra wajah. Perhitungan dimulai dengan memotong citra sesuai dengan ukuran filter, pemotongan tersebut dimulai dari koordinat (1,1), hasil dari pemotongan tersebut ditunjukkan pada gambar 4.12, sedangkan nilai perbedaan

antara *template* wajah dengan potongan citra dapat dilihat pada gambar 4.13. Kemudian dilakukan penjumlahan secara vertikal dan kemudian penjumlahan secara horisontal terhadap nilai perbedaan tersebut untuk disimpan dalam array ukuran 7x3 pada koordinat (1,1) (gambar 4.14), ukuran array 7x3 menggambarkan kemungkinan posisi dari *template* wajah nomor 9 pada citra wajah ukuran 19x19, rumus yang digunakan untuk menghitung ukuran array dapat dilihat pada rumus 4.1 dan 4.2. Kemudian perhitungan total perbedaan nilai antara *template* wajah dengan citra wajah dilanjutkan pada kolom berikutnya yang berada pada koordinat (1,2) hingga koordinat (1,3) yang kemudian dilanjutkan baris berikutnya hingga semua bagian dari citra wajah diproses, sehingga didapatkan total perbedaan nilai yang dapat dilihat pada gambar 4.15. Proses ini kemudian dilakukan terhadap 1999 citra yang lain sehingga didapatkan total nilai perbedaan untuk 2000 citra wajah yang ditunjukkan pada gambar 4.16. Berdasarkan gambar 4.16 dapat dilihat bahwa nilai minimal berada pada koordinat (1,2) (dengan latar berwarna biru), sehingga posisi *template* wajah tersebut pada citra wajah adalah pada baris pertama dan kolom kedua seperti yang ditunjukkan pada gambar 4.17. Dengan menggunakan koordinat tersebut dihitung nilai *threshold awal*, dimana rata-rata dari 2000 citra wajah adalah sebesar 19767 sedangkan rata-rata dari 4000 citra bukan wajah adalah sebesar 26182, sehingga nilai *threshold* awal adalah 22974.5 (dihitung dengan rumus 4.3). Selanjutnya dengan menggunakan nilai *threshold* tersebut dilakukan pencarian nilai *threshold* yang optimal. Pencarian nilai *threshold* yang optimal dimaksudkan untuk mencari nilai *threshold* yang memiliki *false positive* dan *true negative* paling kecil. Pencarian nilai *threshold* yang optimal dilakukan dengan mengimplementasikan tahap 3, hasil dari pencarian nilai *threshold* yang optimal dapat dilihat pada tabel 4.4, berdasarkan tabel 4.4 diketahui bahwa posisi *threshold* yang optimal berada pada posisi kedelapan (dengan latar berwarna biru), posisi kedelapan tersebut merupakan posisi dimana *true negative* yang semakin turun namun *false positive* yang mulai beranjak naik sehingga posisi tersebut adalah posisi dengan kesalahan deteksi yang minimal, perubahan *false positive* dapat dilihat pada gambar 4.18 bagian a (lingkaran warna hijau merupakan posisi kedelapan), sedangkan *true negative* dapat dilihat pada gambar 4.18 bagian b (lingkaran warna hijau merupakan posisi kedelapan). Sedangkan untuk nilai minimal, maksimal,standar deviasi baik untuk

citra wajah maupun bukan wajah dapat dilihat pada tabel 4.5. Sehingga disimpulkan bahwa *template* wajah nomor 9 memiliki posisi di baris pertama dan kolom kedua dan memiliki *threshold* sebesar 22936.

255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	0	0	255	0	0	255	255	255	255	255	255

Gambar 4.9 Bentuk *template* wajah nomor 9

7	46	57	66	71	81	90	96	104	105	105	109	114	117	119	118	119	116	110
39	48	46	38	38	48	65	82	95	99	102	106	106	107	108	108	109	107	110
43	31	11	8	8	1	4	27	53	82	91	86	63	35	24	29	31	45	71
46	31	24	28	29	22	10	12	27	50	69	50	32	11	17	37	51	41	32
55	33	17	6	13	0	8	4	21	57	80	70	28	21	20	17	32	42	46
60	36	23	26	38	26	29	11	24	69	105	77	39	29	37	12	37	38	51
66	53	45	36	42	47	33	26	32	71	102	85	57	43	49	59	73	73	80
72	65	68	7	66	61	48	38	41	69	98	89	82	72	67	70	84	97	104
73	61	70	79	79	69	55	45	50	73	95	88	89	96	101	104	109	112	111
75	52	61	71	73	67	55	43	50	81	96	91	82	98	107	110	112	112	109
101	51	51	65	67	59	46	32	22	42	48	50	69	78	101	109	109	109	101
101	67	50	59	61	51	40	24	18	22	35	32	55	80	87	98	104	101	92
134	89	59	53	57	46	40	38	39	47	59	75	81	91	85	86	96	92	82
212	90	70	52	51	42	42	20	63	76	77	90	87	81	82	85	91	82	76
226	91	78	53	42	40	40	42	43	62	63	68	80	80	78	80	85	76	72
236	112	78	66	40	32	13	3	8	17	19	12	21	41	61	73	82	72	75
249	125	75	75	46	48	31	20	31	45	51	51	45	60	77	80	80	75	81
255	175	76	80	53	46	45	37	43	53	60	72	85	92	94	85	77	76	84
254	199	78	79	66	41	45	45	40	37	51	77	95	97	91	84	75	77	78

Gambar 4.10 Citra wajah dan nilai intensitas

31	85	113	131	145	182	201	213	226	228	228	238	244	246	248	247	248	245	240
58	90	85	56	56	90	127	187	210	218	223	230	230	232	233	233	238	232	240
74	39	10	7	7	1	4	31	107	187	206	195	125	47	27	35	39	79	145
85	39	27	32	35	23	8	12	31	95	139	95	44	10	16	52	102	66	44
110	45	16	4	13	1	7	4	21	113	178	142	32	21	19	16	44	71	85
119	48	24	29	56	29	35	10	27	139	228	165	58	35	52	12	52	56	102
131	107	79	48	71	86	45	29	44	145	223	194	113	74	90	117	153	153	178
148	127	136	134	131	122	90	56	66	139	217	199	187	148	134	142	189	215	226
153	122	142	172	172	139	110	79	95	153	210	197	199	213	222	226	238	244	241
158	103	122	145	153	134	110	74	95	182	213	206	187	217	232	240	244	244	238
222	102	102	127	134	117	85	44	23	71	90	95	139	170	222	238	238	238	222
222	134	95	117	122	102	62	27	17	23	47	44	110	178	196	217	226	222	208
249	199	117	107	113	85	62	56	58	86	117	158	182	206	194	195	213	208	187
251	201	142	103	102	71	71	95	125	161	165	201	196	182	187	194	206	187	161
252	206	170	107	71	62	62	71	74	123	125	136	178	178	170	178	194	161	148
253	244	170	131	62	44	13	2	7	16	18	12	21	66	122	153	187	148	158
254	249	158	158	85	90	39	19	39	79	102	102	79	119	165	178	178	158	182
255	250	161	178	107	85	79	52	74	107	119	148	194	208	208	194	165	161	189
255	251	170	172	131	66	79	66	62	52	102	165	210	215	206	189	158	165	170

Gambar 4.11 Citra dan nilai intensitas hasil *histogram equalization* dari gambar 4.10

31	85	113	131	145	182	201	213	226	228	228	238	244	246	248	247	248
58	90	85	56	56	90	127	187	210	218	223	230	230	232	233	233	238
74	39	10	7	7	1	4	31	107	187	206	195	125	47	27	35	39
85	39	27	32	35	23	8	12	31	95	139	95	44	10	16	52	102
110	45	16	4	13	1	7	4	21	113	178	142	32	21	19	16	44
119	48	24	29	56	29	35	10	27	139	228	165	58	35	52	12	52
131	107	79	48	71	86	45	29	44	145	223	194	113	74	90	117	153
148	127	136	134	131	122	90	56	66	139	217	199	187	148	134	142	189
153	122	142	172	172	139	110	79	95	153	210	197	199	213	222	226	238
158	103	122	145	153	134	110	74	95	182	213	206	187	217	232	240	244
222	102	102	127	134	117	85	44	23	71	90	95	139	170	222	238	238
222	134	95	117	122	102	62	27	17	23	47	44	110	178	196	217	226
249	199	117	107	113	85	62	56	58	86	117	158	182	206	194	195	213

Gambar 4.12 Pemotongan citra hasil *histogram equalization* dengan ukuran 13x17 pixel pada koordinat (1,1)

224	170	142	124	110	73	54	42	29	27	27	17	11	9	7	8	7
197	165	170	199	199	165	128	68	45	37	32	25	25	23	22	22	17
181	39	10	7	7	1	4	224	148	68	206	195	125	47	27	35	216
170	39	27	32	35	23	8	243	224	160	139	95	44	10	16	52	153
145	45	16	4	13	1	7	251	234	142	178	142	32	21	19	16	211
136	48	24	29	56	29	35	245	228	116	228	165	58	35	52	12	203
124	107	79	48	71	86	45	226	211	110	223	194	113	74	90	117	102
107	128	119	121	124	133	165	199	189	116	38	56	68	107	121	113	66
102	133	113	83	83	116	145	176	160	102	45	58	56	42	33	29	17
97	152	133	110	102	121	145	181	160	73	42	49	68	38	23	15	11
33	153	153	128	121	138	170	211	232	184	165	160	116	85	33	17	17
33	121	160	138	133	153	193	228	238	232	208	211	145	77	59	38	29
6	56	138	148	142	170	62	56	197	86	117	97	73	49	61	60	42

Gambar 4.13 Nilai perbedaan antara *template* wajah nomor 9 dengan gambar 4.12

21881		

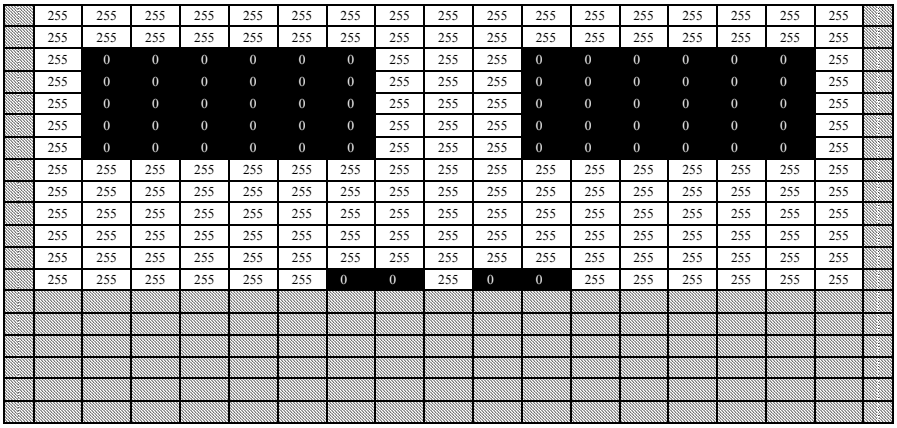
Gambar 4.14 Total nilai perbedaan dari gambar 4.13 yang disimpan dalam array ukuran 7x3 pada koordinat (1,1)

21881	19952	18442
24952	23505	22124
28377	27155	26048
30469	29641	28704
31301	31224	30739
30093	30537	30691
28482	29177	29484

Gambar 4.15 Total nilai perbedaan dari gambar 4.13 untuk semua koordinat

41903965	**39352404**	40281381
47193311	44858085	45702111
54569280	52579014	53324706
61024216	59408209	60089999
64563094	63317686	63928211
64179479	63144273	63672701
61692609	60827904	61241783

Gambar 4.16 Total nilai perbedaan antara *template* wajah nomor 9 dengan 2000 citra wajah



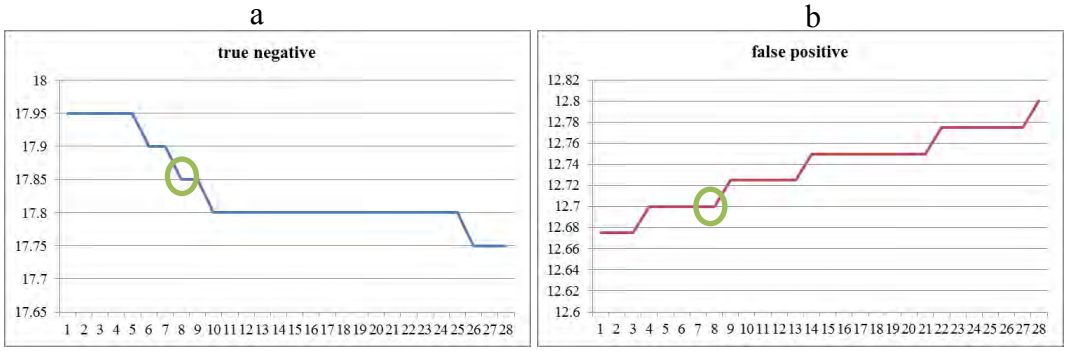
Gambar 4.17 Posisi *template* wajah nomor 9 pada bidang 19x19 piksel

Tabel 4.4 Hasil pencarian *threshold* yang optimal untuk *template* wajah nomor 9

No.	True Negative	False Positive	sumErr	Threshold
1	17.95	12.675	14.433	22929
2	17.95	12.675	14.433	22930
3	17.95	12.675	14.433	22931
4	17.95	12.7	14.45	22932
5	17.95	12.7	14.45	22933
6	17.9	12.7	14.433	22934
7	17.9	12.7	14.433	22935
8	**17.85**	**12.7**	**14.417**	**22936**
9	17.85	12.725	14.433	22937
10	17.8	12.725	14.417	22938
11	17.8	12.725	14.417	22939
12	17.8	12.725	14.417	22940
13	17.8	12.725	14.417	22941
14	17.8	12.75	14.433	22942
15	17.8	12.75	14.433	22943
16	17.8	12.75	14.433	22944
17	17.8	12.75	14.433	22945
18	17.8	12.75	14.433	22946
19	17.8	12.75	14.433	22947
20	17.8	12.75	14.433	22948
21	17.8	12.75	14.433	22949
22	17.8	12.775	14.45	22950
23	17.8	12.775	14.45	22951

No.	True Negative	False Positive	sumErr	Threshold
24	17.8	12.775	14.45	22952
25	17.8	12.775	14.45	22953
26	17.75	12.775	14.433	22954
27	17.75	12.775	14.433	22955
28	17.75	12.8	14.45	22956

a b



Gambar 4.18 Grafik *true negative* dan *false positive*

Tabel 4.5 Nilai *template* wajah nomor 9

	Wajah	Bukan wajah
Minimal	13638	16091
Maksimal	31800	38211
Rata-rata	19676	26182
Standar deviasi	3185.4	2847.8

4.1.5.2 Tahap Deteksi

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 4.19 adalah sebagai berikut:

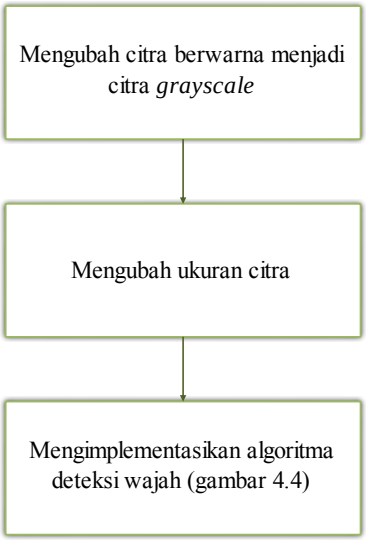
1. **Mengubah citra berwarna menjadi citra *grayscale***

   Mengubah citra berwarna menjadi citra *grayscale* dengan rumus 4.6.

$$Y = 0.299\ Red + 0.587\ Green + 0.114\ Blue \quad (4.6)$$

2. **Mengubah ukuran citra**

Mengubah ukuran citra menjadi sebesar 40x30 piksel dengan menggunakan fungsi “imresize” yang telah disediakan oleh matlab.



Gambar 4.19 Alur tahap deteksi wajah

3. **Mengimplementasikan algoritma deteksi wajah yang terdapat pada lampiran B.**

Proses identifikasi wajah dilakukan dengan cara mengambil potongan citra atau yang biasa disebut sebagai window dengan ukuran 19x19. 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 4.19 dapat dilihat pada gambar 4.20. Implementasi tersebut diawali dengan membaca citra berwarna (gambar 4.20 bagian a), kemudian mengubahnya menjadi citra *grayscale* (gambar 4.20 bagian b), kemudian mengubah ukuran citra *grayscale* menjadi citra *grayscale* yang berukuran 40x30 piksel (gambar 4.20 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 4.20 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 4.20 bagian e), sehingga langkah terakhir dicari nilai identifikasi wajah yang paling kecil (gambar 4.20 bagian f).

a b c d e f

Gambar 4.20 Hasil implementasi 4.19

4.1.6 Uji Kebenaran dan Uji Kinerja

Uji kebenaran dan uji kinerja yang dijelaskan pada sub bab ini adalah uji kebenaran dan uji kinerja yang dilakukan pada saat algoritma deteksi wajah diimplementasikan dengan menggunakan komputer. Pada sub bab ini dijelaskan mengenai skenario yang digunakan, hasil uji coba baik uji kebenaran maupun uji kinerja, dan analisa dari hasil uji coba.

4.1.6.1 Skenario Uji Coba

Pada bagian ini, skenario uji coba 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:

1. 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.

2. 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°.

Kecepatan dari algoritma deteksi wajah dalam mengidentifikasi wajah dihitung pada saat melakukan kedua skenario tersebut.

4.1.6.2 Hasil Uji Kebenaran

Pada tahap ini uji kebenaran diawali dengan pemilihan *template* wajah terlebih dahulu dengan menggunakan skenario pemilihan *threshold*. Kemudian dilanjutkan dengan skenario posisi wajah miring.

4.1.6.2.1 Skenario Pemilihan Threshold

Skenario pemilihan *threshold* dilakukan dengan berdasarkan pada hasil *training*. Hasil training dari penelitian ini dapat dilihat pada lampiran C. Dari lampiran C, diketahui ada dua *template* wajah yang memiliki *false positive* dan *true negative* yang rendah yaitu *template* wajah nomor 32 dan *template* wajah nomor 34. *Template* wajah nomor 32 memiliki *false positive* sebesar 12.45%, dan *true negative* sebesar 9.05%. Sedangkan *template* wajah nomor 34 memiliki *false positive* sebesar 11,78% dan *true negative* sebesar 9,75%.

Karena berdasarkan hasil *training* diperoleh dua *template* wajah, maka perlu dilakukan perbandingan *detection rate* dan *false positive* dari kedua *template* wajah tersebut. *Template* wajah yang menghasilkan *detection rate* dan *false positive* yang lebih baik akan digunakan untuk melakukan deteksi wajah. Untuk memilih *template* wajah mana yang akan digunakan untuk melakukan deteksi wajah, perlu dilakukan uji coba dengan menggunakan skenario pemilihan *threshold*. Masing-masing *template* wajah akan diuji coba dengan berdasarkan pada nilai *threshold*, minimal, maksimal, rata-rata, dan standar deviasi. Uji coba akan dilakukan dengan menggunakan 100 foto orang indonesia. Berikut ini adalah proses pemilihan *template* wajah:

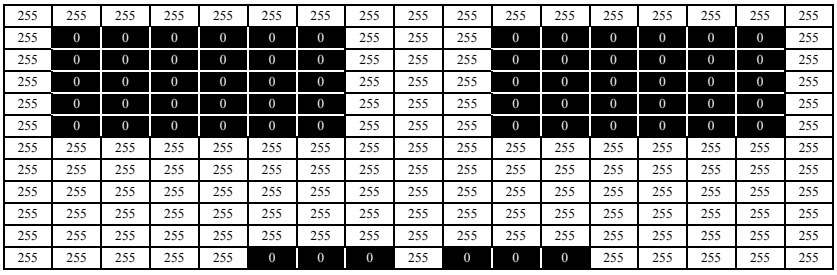
- **Template wajah nomor 32**

  Bentuk *template* wajah nomor 32 dapat dilihat pada gambar 4.21. Posisi *template* wajah tersebut pada window dapat dilihat pada gambar 4.22, *template* wajah tersebut berada pada baris ke-1 dan kolom ke-2. *Threshold* dari *template* wajah tersebut adalah sebesar 21054, sedangkan untuk nilai minimal, maksimal, rata-rata, dan standar deviasi untuk data wajah dan bukan wajah dapat dilihat pada tabel 4.6. Hasil skenario pemilihan *threshold* dapat dilihat pada tabel 4.7.

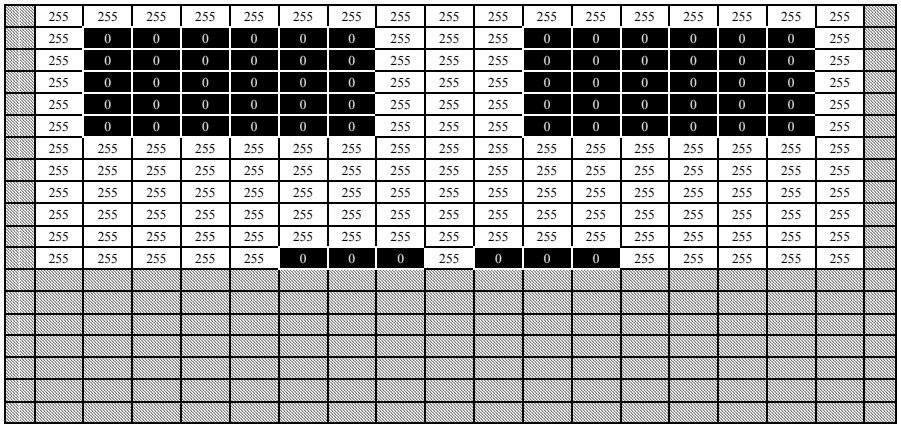
- **Template wajah nomor 34**

  Gambar 4.23 merupakan bentuk *template* wajah nomor 34, sedangkan gambar 4.24 adalah posisi *template* wajah tersebut pada window, dimana *template* wajah tersebut berada pada baris ke-1 dan kolom ke-2. *Template* wajah nomor 34 memiliki *threshold* sebesar 23006, sedangkan untuk nilai

minimal, maksimal, rata-rata, dan standar deviasi untuk data wajah dan bukan wajah dapat dilihat pada tabel 4.8. Untuk hasil dari skenario pemilihan *threshold* dapat dilihat pada tabel 4.9.



Gambar 4.21 *Template* wajah nomor 32



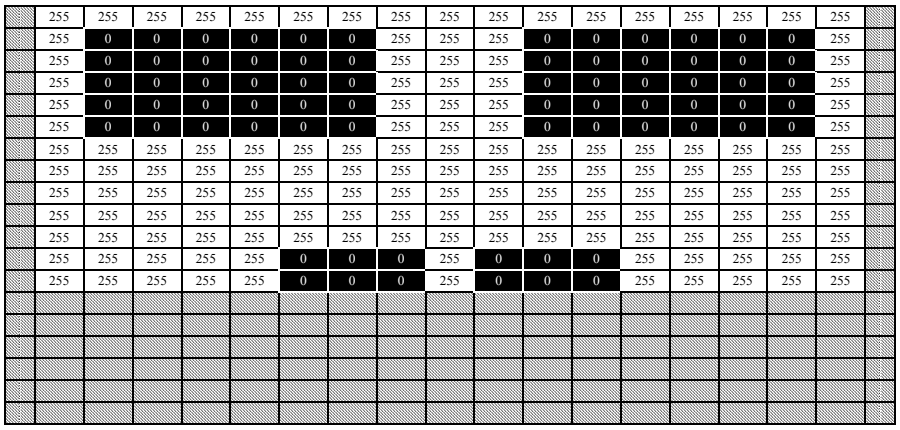
Gambar 4.22 Posisi *template* wajah nomor 32 pada window

Tabel 4.6 Nilai *template* wajah nomor 32

	Wajah	Bukan wajah
Minimal	12707.0	15411.0
Maksimal	28448.0	33698.0
Rata-rata	17983.0	24262.0
Standar deviasi	2253.1	2716.0

Tabel 4.7 Skenario pemilihan *threshold* untuk *template* wajah nomor 32

No	*Threshold*	*Detection Rate*	*False Positive*
1	*Threshold* > 12707 & *Threshold* < 15411	27%	2,65%
2	***Threshold* > 12707 & *Threshold* < 17983**	**100%**	**0,38% - 10,61%**
3	*Threshold* > 12707 & *Threshold* < 21054	100%	0,38% - 23,48%
4	*Threshold* > 15730 & *Threshold* < 21054	100%	0,38% - 23,48%



255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	0	0	0	255	0	0	0	255	255	255	255	255
255	255	255	255	255	0	0	0	255	0	0	0	255	255	255	255	255

Gambar 4.23 *Template* wajah nomor 34

.

▨	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	▨
▨	255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255	▨
▨	255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255	▨
▨	255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255	▨
▨	255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255	▨
▨	255	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	255	▨
▨	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	▨
▨	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	▨
▨	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	▨
▨	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	▨
▨	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	▨
▨	255	255	255	255	255	0	0	0	255	0	0	0	255	255	255	255	255	▨
▨	255	255	255	255	255	0	0	0	255	0	0	0	255	255	255	255	255	▨
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Gambar 4.24 Posisi *template* wajah nomor 34 pada window

Tabel 4.8 Nilai *template* wajah nomor 34

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

Tabel 4.9 Skenario pemilihan *threshold* untuk *template* wajah nomor 34

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%

Berdasarkan tabel 4.7, posisi *threshold* dari *template* wajah nomor 32 yang menghasilkan *detection rate* dan *false positive* yang paling baik berada diantara

12707 dan 17983 (warna latar biru). Pada posisi tersebut *detection rate* yang diperoleh sebesar 100%, sedangkan *false positive* sebesar 0,38% hingga 10,61%. Sedangkan jika melihat pada tabel 4.9, posisi *threshold* yang memiliki *detection rate* dan *false positive* yang paling baik berada diantara 14173 dan 19892 (warna latar biru). *Detection rate* pada posisi tersebut sebesar 100%, sedangkan *false* positive dari posisi tersebut sebesar 0.38% - 9.47%. Jika melihat dari nilai *detection rate* dan *false positive* yang diperoleh dari *template* wajah nomor 32 maupun *template* wajah nomor 34, dapat diambil kesimpulan bahwa *template* wajah nomor 34 memiliki kemampuan identifikasi wajah yang lebih baik dari pada *template* wajah nomor 32, karena *template* wajah nomor 34 memiliki *false positive* yang lebih kecil. Sehingga *template* wajah yang digunakan untuk melakukan proses deteksi wajah adalah *template* wajah nomor 34.

4.1.6.2.2 Skenario Posisi Wajah Miring

Langkah selanjutnya setelah diketahui bahwa *template* wajah nomor 34 merupakan *template* wajah yang paling optimal untuk melakukan deteksi wajah adalah menguji coba *template* wajah tersebut dengan skenario posisi wajah miring. Uji coba ini dilakukan dengan menggunakan 50 foto orang indonesia. Hasil uji coba dari skenario posisi wajah miring dapat dilihat pada tabel 4.10.

Tabel 4.10 Skenario posisi wajah miring

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%

4.1.6.3 Hasil Uji Kinerja

Uji kinerja pada bagian ini 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

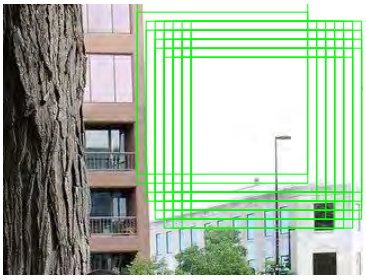
ukuran 320x240 piksel. Berdasarkan hasil uji coba, semakin banyak bagian yang teridentifikasi sebagai wajah, maka kecepatan yang dihasilkan juga akan semakin lambat. Selama proses uji coba, kecepatan algoritma untuk melakukan identifikasi wajah adalah sebesar 7-18 fps.

4.1.6.4 Analisa Hasil

Berdasarkan hasil uji coba yang telah dilakukan pada sub bab 4.1.6.2 dan 4.1.6.3 dapat dinyatakan bahwa algoritma yang diusulkan dalam penelitian ini memiliki kemampuan identifikasi wajah yang baik dan cepat, sehingga algoritma ini dapat diimplementasikan pada arduino. Berikut ini adalah penjelasan dari berbagai sudut pandang mengenai kemampuan identifikasi wajah dari algoritma yang diusulkan dalam penelitian ini:

- Dari sisi *false positive*, kesalahan identifikasi keberadaan wajah pada citra bukan wajah adalah sebesar 0.38% hingga 10%. Contoh *false positive* yang terjadi dapat dilihat pada gambar 4.25. Posisi baris dan kolom serta nilai fitur dari setiap *false positive* pada gambar 4.25 dapat dilihat pada tabel 4.11. Dari gambar dan tabel tersebut dapat dilihat bahwa algoritma ini tidak cocok untuk diimplementasikan di lingkungan luar (*outdoor*).
- Dari sisi keberhasilan mendeteksi wajah, algoritma ini memiliki *detection rate* sebesar 98%-100%. Berdasarkan tabel 4.10, algoritma ini masih memiliki kemampuan identifikasi wajah yang cukup baik walaupun posisi wajah miring hingga 5°. Contoh keberhasilan dalam mengidentifikasi wajah untuk posisi wajah frontal dapat dilihat pada gambar 4.26. Sedangkan untuk posisi wajah miring dengan kemiringan 1° dapat dilihat pada gambar 4.27, untuk kemiringan 2° dapat dilihat pada gambar 4.28, untuk kemiringan 5° ditunjukkan pada gambar 4.29, dan untuk kemiringan sebesar 10° dapat dilihat pada gambar 4.30. Bagian A dari gambar-gambar tersebut menunjukkan semua bagian dari citra yang teridentifikasi sebagai wajah, yang ditunjukkan oleh persegi yang berwarna hijau. Sedangkan bagian B dari gambar-gambar tersebut menunjukkan bagian dari citra yang teridentifikasi sebagai wajah yang memiliki nilai fitur paling minimal, nilai fitur paling minimal ditunjukkan pada tabel dengan warna latar biru. Posisi baris dan

kolom yang teridentifikasi sebagai wajah dan nilai fitur dari gambar 4.26 dapat dilihat pada tabel 4.12, sedangkan untuk gambar 4.27 dapat dilihat pada tabel 4.13, untuk gambar 4.28 ditunjukkan oleh tabel 4.14, sedangkan gambar 4.29 dapat dilihat pada tabel 4.15, dan untuk gambar 4.30 dapat dilihat pada tabel 4.16.

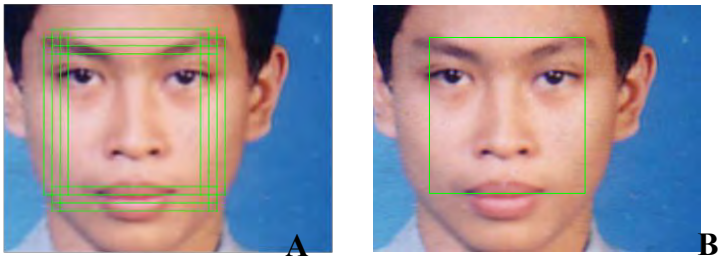


Gambar 4.25 Contoh *false positive* dari algoritma deteksi wajah yang diimplementasikan pada komputer

Tabel 4.11 Hasil deteksi dari gambar 4.25

No.	Baris	Kolom	Nilai Fitur
1	1	16	19049
2	2	16	19266
3	3	17	19091
4	3	18	19096
5	3	19	18596
6	3	20	18098
7	3	21	17632
8	3	22	18127
9	4	17	19762
10	4	18	19323
11	4	19	18358
12	4	20	17416
13	4	21	17412
14	4	22	17927
15	5	18	19295
16	5	19	18385
17	5	20	17947
18	5	21	18775

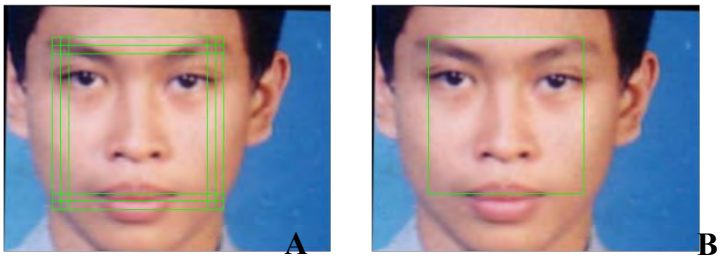
No.	Baris	Kolom	Nilai Fitur
19	5	22	18843
20	6	18	19659
21	6	19	18798
22	6	20	18797
23	6	21	19564
24	6	22	19624
25	7	19	19150
26	7	20	19143
27	7	21	19855



Gambar 4.26 Contoh uji coba dengan posisi wajah frontal

Tabel 4.12 Hasil deteksi dari gambar 4.26

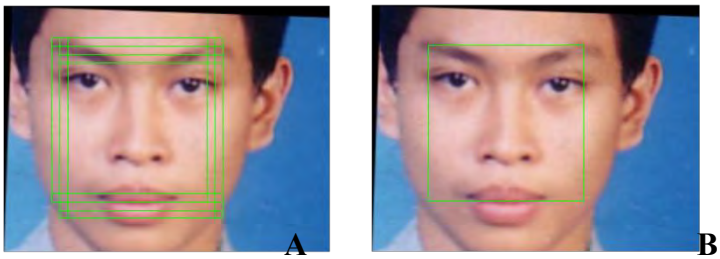
No.	Baris	Kolom	Nilai Fitur
1	4	7	19856
2	4	8	19731
3	5	6	19696
4	5	7	18056
5	**5**	**8**	**18033**
6	5	9	19380
7	6	7	18839
8	6	8	18664
9	7	7	18854
10	7	8	18592



Gambar 4.27 Contoh uji coba dengan posisi kemiringan wajah sebesar 1°

Tabel 4.13 Hasil deteksi dari gambar 4.27

No.	Baris	Kolom	Nilai Fitur
1	5	7	18631
2	**5**	**8**	**17958**
3	5	9	18949
4	6	7	19228
5	6	8	18525
6	6	9	19378
7	7	7	19465
8	7	8	18553
9	7	9	19459

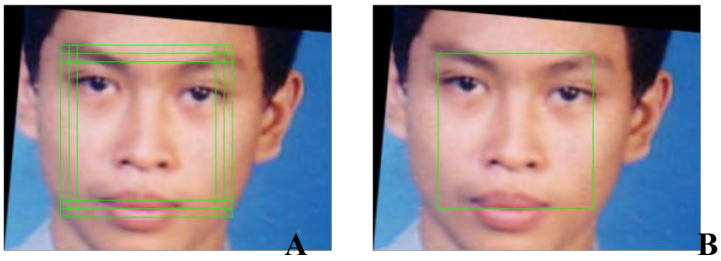


Gambar 4.28 Contoh uji coba dengan posisi kemiringan wajah sebesar 2°

Tabel 4.14 Hasil deteksi dari gambar 4.28

No.	Baris	Kolom	Nilai Fitur
1	5	7	19695
2	5	8	18518
3	5	9	19090
4	6	7	19427
5	**6**	**8**	**18163**

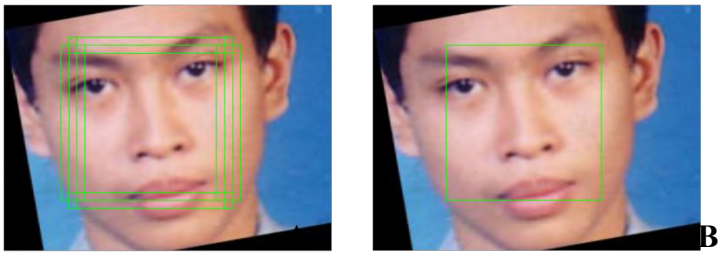
No.	Baris	Kolom	Nilai Fitur
6	6	9	18631
7	7	8	18598
8	7	9	18995
9	8	8	19113
10	8	9	19396



Gambar 4.29 Contoh uji coba dengan posisi kemiringan wajah sebesar 5°

Tabel 4.15 Hasil deteksi dari gambar 4.29

No.	Baris	Kolom	Nilai Fitur
1	6	8	19690
2	6	9	18879
3	6	10	19496
4	7	8	19497
5	**7**	**9**	**18451**
6	7	10	18992
7	8	8	19784
8	8	9	18625
9	8	10	19404



Gambar 4.30 Contoh uji coba dengan posisi kemiringan wajah sebesar 10°

Tabel 4.16 Hasil deteksi dari gambar 4.30

No.	Baris	Kolom	Nilai Fitur
1	5	9	19413
2	5	10	19124
3	6	8	19841
4	6	9	18529
5	**6**	**10**	**18357**
6	6	11	19502
7	7	9	19576
8	7	10	19498

- Dari sisi kecepatan, algoritma ini mampu mengidentifikasi citra yang memiliki ukuran 320x240 piksel dengan kecepatan sebesar 7-18 fps.

4.2 Pengimplementasian 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 4.31. 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.

Urutan proses deteksi wajah dapat dilihat pada gambar 3.3. Secara garis besar urutan proses tersebut dijelaskan sebagai berikut:

1. **Pengambilan obyek**

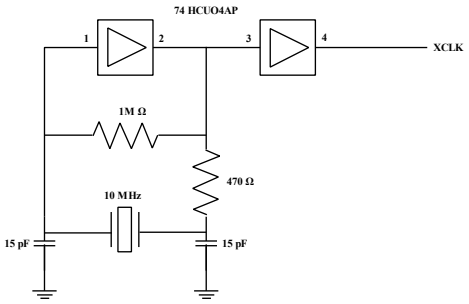
   Proses pengambilan obyek dilakukan oleh serial kamera ov7670 *without FIFO*. Agar serial kamera dapat berjalan, serial kamera tersebut dihubungkan dengan arduino. Untuk koneksi antara serial kamera dengan arduino dapat dilihat pada tabel 4.17. Khusus untuk pin XCLK, pin tersebut dihubungkan dengan *system clock*. Desain *system clock* pada penelitian ini dapat dilihat pada gambar 4.32. Untuk proses pengambilan obyek dari serial kamera ov7670 *without FIFO* dijelaskan pada gambar 4.33 dan gambar 4.34. Hasil dari proses pengambilan obyek adalah citra, dimana citra tersebut nantinya

akan diolah dengan menggunakan algoritma deteksi wajah. Berdasarkan gambar 4.33 dan gambar 4.34, 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 4.34). 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 4.33). 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.

Tabel 4.17 Koneksi pin ov7670 *without FIFO* dengan pin arduino

No.	PIN ov7670 *without FIFO*	TIPE	KETERANGAN	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*

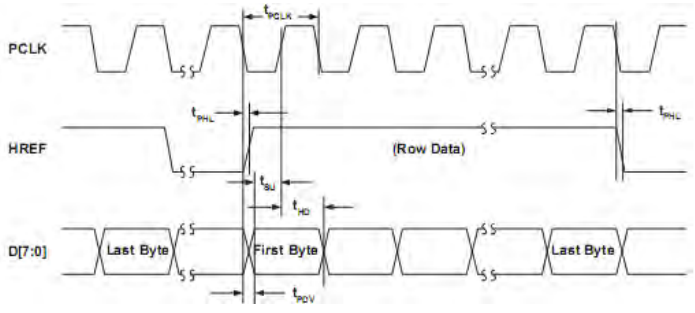
Gambar 4.31 Sistem deteksi wajah yang dibangun dengan arduino Mega 2560, serial kamera ov7670 *without FIFO*, dan servo S3003



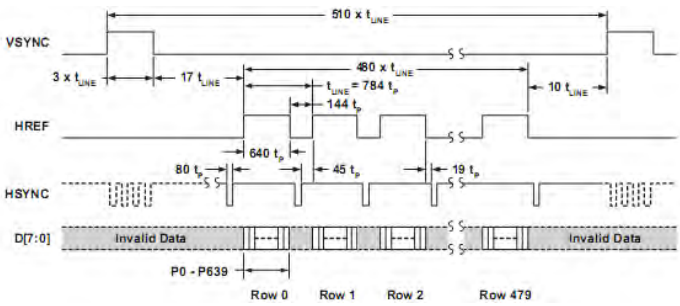
Gambar 4.32 Desain *system clock*

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 lampiran B. 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.

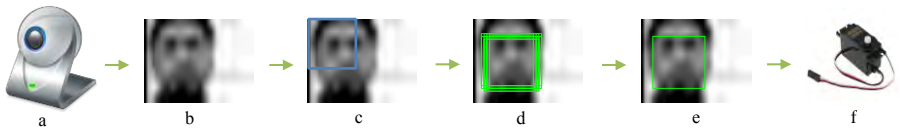


Gambar 4.33 *Horizontal timing* (Omnivision, 2006)



Gambar 4.34 *Frame Timing* (Omnivision, 2006)

Hasil implementasi dari langkah-langkah yang terdapat pada gambar 3.3 dapat dilihat pada gambar 4.35. Sistem deteksi wajah pada arduino diawali dari pengambilan obyek oleh serial kamera ov7670 *without FIFO* (gambar 4.35 bagian a), dimana hasil dari pengambilan obyek dapat dilihat pada gambar 4.35 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 4.35 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 4.35 bagian d), kemudian dicari nilai identifikasi wajah yang paling kecil (gambar 4.35 bagian e), selanjutnya dengan menggunakan koordinat dari nilai identifikasi wajah yang paling kecil arduino menggerakkan servo menyesuaikan koordinat tersebut sehingga untuk pengambilan obyek berikutnya posisi wajah berada di tengah kamera (gambar 4.35 bagian f).



Gambar 4.35 Hasil implementasi sistem deteksi wajah pada arduino

4.3 Uji Kebenaran dan Uji Kinerja

Uji kebenaran dan uji kinerja yang dijelaskan pada sub bab ini merupakan uji kebenaran dan uji kinerja dari algoritma deteksi wajah yang

diimplementasikan pada arduino. Pada sub bab ini dibahas mengenai skenario uji coba, hasil uji kebenaran, hasil uji kinerja, dan analisa dari hasil uji coba.

4.3.1 Skenario dan Lingkungan Uji Coba

Pada sub bab ini dijelaskan mengenai skenario uji coba dan lingkungan yang digunakan untuk menjalankan skenario.

4.3.1.1 Skenario Uji Coba

Skenario uji coba pada sub bab ini 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:

1. 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. Uji coba pada bagian ini dilakukan dengan menggunakan 10 orang indonesia dengan 3 posisi wajah frontal dan 2 posisi wajah miring.
2. 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.
3. 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.

4.3.1.2 Lingkungan Uji Coba

Uji coba pada penelitian ini dilakukan pada lingkungan yang berbeda-beda. Dimana, kondisi lingkungan yang digunakan dalam penelitian ini dapat dilihat pada tabel 4.18.

Tabel 4.18 Kondisi lingkungan uji coba

No.	Jenis Ruangan	Ukuran Ruangan (m)	Ukuran Jendela (cm)	Daya Lampu (watt)	Waktu Uji Coba
1	Ruangan tertutup	3x3	1,25x0,5	11	20.00-21.00
2	Ruangan terbuka	6x3	2x2	-	15.00-17.30
3	Ruangan terbuka	6x3	2x2	30	18.00-19.00
4	Ruangan tertutup	4x3	1,25x0.5	20	12.30-17.00
5	Ruangan tertutup	4x3	2x2	8	10.00-11.00

4.3.2 Hasil Uji Kebenaran

Uji kebenaran dilakukan dengan menggunakan tiga skenario yang telah dijelaskan pada sub bab 4.3.1.1. Hasil dari skenario posisi wajah frontal dan miring dapat dilihat pada lampiran D, sedangkan skenario mendeteksi koordinat wajah dapat dilihat pada lampiran E, dan skenario wajah yang bergerak dapat dilihat pada lampiran F. Sedangkan lingkungan uji coba yang digunakan dijelaskan pada sub bab 4.3.1.2. Berdasarkan tabel 4.18, Uji coba pada orang pertama menggunakan ruangan nomor lima, sedangkan orang kedua, keempat, kelima, keenam, dan ketujuh menggunakan ruangan nomor empat, orang ketiga menggunakan ruangan nomor satu, orang ke 8 menggunakan ruangan nomor tiga, dan orang kesembilan dan kesepuluh menggunakan ruangan nomor dua. Dimana, urutan orang pertama hingga kesepuluh menggunakan urutan dari Lampiran D, E,

dan F. Hasil skenario tersebut dapat dilihat pada tabel 4.19. Sedangkan *false positive* yang terjadi selama tahap uji coba adalah sebesar 0,42% - 14,167%.

Tabel 4.19 Skenario uji coba pada 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%

4.3.3 Hasil Uji Kinerja

Uji kinerja pada sub bab ini 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.

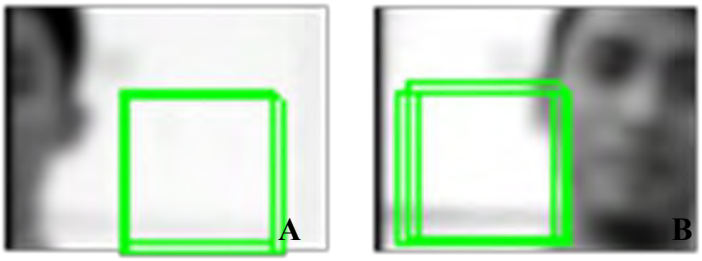
4.3.4 Analisa Hasil

Berdasarkan uji coba yang telah dilakukan pada sub bab 4.3.2 dan 4.3.3 dapat dinyatakan bahwa arduino Mega 2560 tetap dapat menjalankan algoritma deteksi wajah yang diusulkan dalam penelitian ini dengan baik. Berikut ini adalah penjelasan dari berbagai sudut pandang mengenai kemampuan arduino dalam melakukan identifikasi wajah:

- Dari sisi *false positive*, kesalahan dalam melakukan identifikasi wajah adalah sebesar 0,42%-14,167%. Besarnya *false positive* dipengaruhi oleh warna latar, jika warna latar adalah putih, maka sangat besar kemungkinan terjadi kesalahan identifikasi wajah. Contoh citra yang terdapat *false positive* dapat dilihat pada gambar 4.36. Nilai fitur serta posisi baris dan kolom dari *false positive* pada gambar 4.36 bagian A dapat dilihat pada tabel 4.20, sedangkan untuk gambar 4.36 bagian B ditunjukkan pada tabel 4.21.

- Dari sisi keberhasilan arduino dalam mendeteksi wajah frontal dan miring, *detection rate* yang berhasil dicapai adalah sebesar 80% - 100%. Keberhasilan identifikasi wajah dipengaruhi oleh jarak wajah dari kamera. Wajah yang besar memerlukan jarak yang lebih jauh daripada wajah yang kecil. Berdasarkan lampiran D, 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 4.36 Contoh false positive dari algoritma deteksi wajah yang diimplementasikan pada arduino

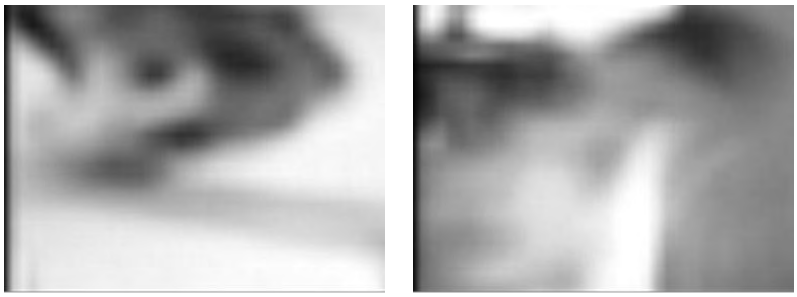
Tabel 4.20 Hasil deteksi dari gambar 4.36 A

No.	Baris	Kolom	Nilai Fitur
1	11	15	19675
2	12	15	18689
3	12	16	19001

Tabel 4.21 Hasil deteksi dari gambar 4.36 B

No.	Baris	Kolom	Nilai Fitur
1	10	5	19747
2	11	4	19488
3	11	5	19130
4	11	6	19539

- Dari sisi keberhasilan arduino dalam mengidentifikasi wajah yang sedang bergerak, *detection rate* yang dicapai adalah sebesar 60% - 100%. Keberhasilan dalam mengidentifikasi wajah yang bergerak dipengaruhi oleh kecepatan kamera dalam menangkap obyek dan pergerakan manusia. Contoh penangkapan obyek yang dilakukan oleh serial kamera ov7670 *without FIFO* pada saat obyeknya sedang bergerak dapat dilihat pada gambar 4.37. Dari gambar tersebut dapat disimpulkan bahwa jika pergerakan terlalu cepat maka obyek yang diambil oleh serial kamera tidak sempurna.
- Dari sisi kecepatan, lamanya proses deteksi wajah dipengaruhi oleh kinerja serial kamera ov7670 *without FIFO* dan kecepatan arduino. Proses pengambilan gambar dengan menggunakan kamera tersebut memerlukan waktu 3-12 detik, dengan ukuran citra sebesar 40x30 piksel. Dan proses pengolahan gambar oleh arduino membutuhkan waktu antara 6 hingga 8 detik.



Gambar 4.37 Contoh citra yang diambil dengan menggunakan arduino

BAB 5
KESIMPULAN DAN SARAN

Dalam bab ini dijelaskan mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan. Selain itu, pada bab ini juga dijelaskan saran yang perlu dilakukan untuk menyempurnakan penelitian ini.

5.1 Kesimpulan

Kesimpulan yang didapat dari hasil uji coba penelitian ini adalah sebagai berikut:

1. Berdasarkan pada sub bab 4.1.2 dan sub bab 4.1.3, metode *template matching* merupakan metode yang paling cocok untuk diimplementasikan pada *embedded system* yang memiliki memori dan kecepatan yang terbatas.
2. Berdasarkan hasil *training* yang terdapat pada lampiran C, diketahui bahwa *template* wajah yang memiliki *detection rate* yang paling baik adalah *template* wajah yang terdiri dari mata dan hidung (*template* wajah nomor 32 dan 34). Bentuk *template* wajah nomor 32 ditunjukkan pada gambar 4.21, sedangkan bentuk *template* wajah nomor 34 ditunjukkan pada gambar 4.23. Dan berdasarkan hasil uji coba yang dijelaskan pada sub bab 4.1.6.2.1 *template* wajah nomor 34 merupakan *template* wajah yang memiliki *detection rate* dan *false positive* yang lebih baik dari pada *template* wajah nomor 32.
3. Berdasarkan hasil uji coba yang telah dilakukan pada sub bab 4.3.2 dapat diketahui bahwa detection rate yang berhasil dicapai pada penelitian ini adalah sebesar 80% hingga 100%

5.2 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.

BIOGRAFI PENULIS

Penulis dilahirkan di Surabaya pada tanggal 21 Agustus 1986 dengan nama Yupit Sudianto. Penulis mendapatkan gelar sarjana komputer dari Jurusan Sistem Informasi FTIf ITS pada tahun 2009. Pada saat menyelesaikan gelar sarjananya penulis mengambil bidang minat PPSI dan menghasilkan tugas akhir yang berjudul "Rancang Bangun Sistem Informasi Penerimaan Siswa Baru Online : Modul Pendaftaran". Bidang yang menjadi minat penulis adalah rekayasa perangkat lunak dan pengolahan citra. Untuk menghubungi penulis dapat dilakukan melalui alamat email ic.yupit@gmail.com.

DAFTAR PUSTAKA

Banzi, M., 2011. "Getting Started with Arduino". O'Reilly.

Chen, C.R., Wong, W.S., dan Chiu, C.T., 2010. "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 Vol 19 No 11.

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

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

Dai, Y., dan Nakano, Y., 1996. Face-Texture Model Based on SGLD and Its Application in Face Detection in a Color Scene. Pattern Recognition vol 29 no 6 pp 1007-1017.

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

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

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

Kamal, R., 2008. Embedded Systems: Architecture, Programming and Design. McGraw-Hill.

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

Kyrkou, C., dan Theocharides, T., 2010. “A Flexible Parallel Hardware Architecture for AdaBoost-Based Real-Time Object Detection”. VLSI, vol 19 pp 1034-1047.

Kyrkou, C., Ttofis, C., dan Theocharides, T., 2011. “FPGA-Accelerated Object Detection Using Edge Information”. FPL, pp 167-170.

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

Lanitis, A., Taylor, C.J., dan Cootes, T.F., 1995. An Automatic Face Identification System Using Flexible Appearance Models. Image and Vision Computing vol 13 no 5 pp 393-401.

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

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

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

McCready, R., 2000. Real-Time Face Detection on a Configurable Hardware System. ACM, FPL ’00 pp 157-162.

McKenna, S., Gong, S., dan Raja, Y., 1998. Modelling Facial Colour and Identify with Gaussian Mixtures. Pattern Recognition vol 31 no 12 pp 1883-1892.

Negru, S., 2010. “A conceptual architecture of an arduino-based social-emotional interactive system”. IEEE, Intelligent Computer Communication and Processing.

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

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

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

Puspita, E., 2004. “Sistem Pendeteksi dan Penjejakan Wajah Secara Real Time”. Fakultas Teknologi Informasi. Institut Teknologi Sepuluh Nopember.

Rajagopalan, A., Kumar, K., Karlekar, J., Manivasakan, R., Patil, M., Desai, U., Poonacha, P., dan Chaudhuri, S.,1988. Finding Faces in Photographs. Proc, Sixth IEEE Intl’l Conf, Computer Vision pp 640-645.

Rowley, H., Baluja, S., dan Kanade, T., 1998. Neural Network-Based Face Detection. IEEE Trans, Pattern Analysis and Machine Intelligence vol 20 no 1 pp 23-38.

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

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

Sung, K.K., dan Poggio, T., 1998. Example-Based Learning for View-Based Human Face Detection. IEEE Trans, Pattern Analysis and Machine Intelligence vol 20 no 1 pp 39-51.

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

Theocharides, T., 2006. “A parallel Architecture for hardware face detection”. IEEE, Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures.

Turk, M., dan Pentland, A., 1991. Eigenfaces for Recognition. I.Cognitive Neuroscience vol 3 no 1 pp 71-86.

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

Yang, G., dan Huang, T.S., 1994. Human Face Detection in Complex Background. Pattern Recognition vol 27 no 1 pp 53-63.

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

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

Yow, K.C., dan Cipolla, R., 1997. Feature-Based Human Face Detection. Image and Vision Computing vol 15 no 9 Pp 713-735.

Yutong, Z., Guosheng, dan Y., Licheng, W., 2010. “Fast Face in Field Programmable Gate Array”. IEEE, International Conference on Digital Manufacturing & Automation.

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

LAMPIRAN A
ALGORITMA TRAINING

Tahap 1 - mencari posisi fitur

- 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.

Tahap 2 - 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.

Tahap 3 – 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.

Halaman ini sengaja dikosongkan

LAMPIRAN B

ALGORITMA DETEKSI 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.

Halaman ini sengaja dikosongkan

LAMPIRAN C

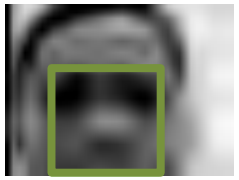
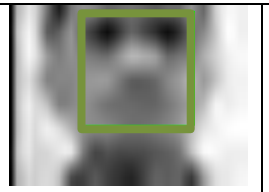
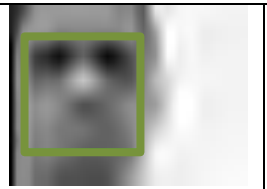
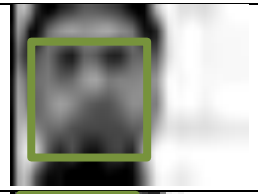
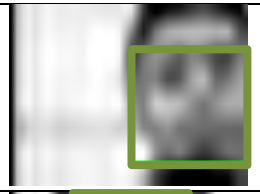
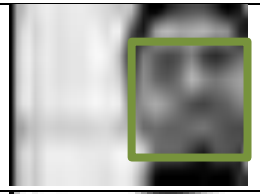
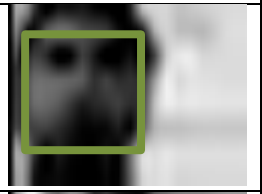
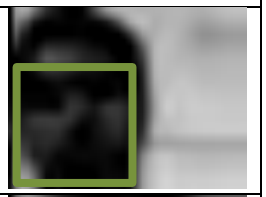
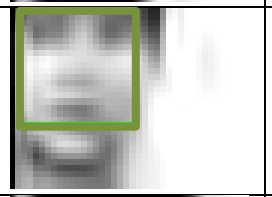
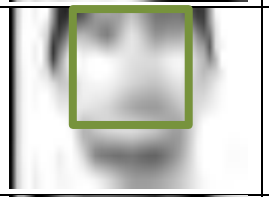
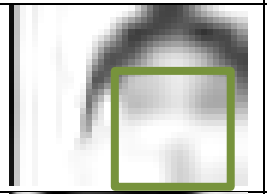
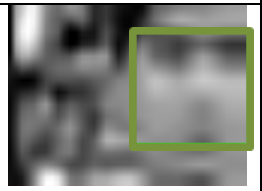
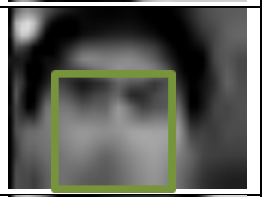
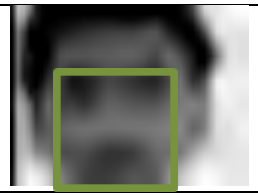
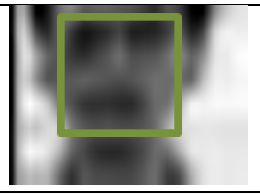
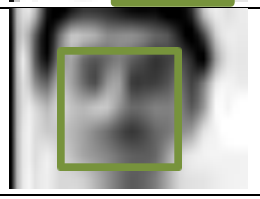
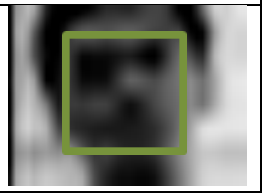
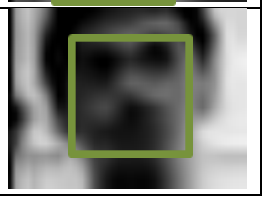
HASIL TRAINING

No. Fitur	posisi baris	panjang baris	posisi kolom	panjang kolom	threshold	true negative	false positive	sumError
1	1	11	2	16	21029	11.05	11.675	11.467
2	1	12	3	14	19896	12.05	14.25	13.517
3	1	12	3	14	19845	14	14.9	14.6
4	1	12	3	14	19941	13.65	15.025	14.567
5	11	1	2	16	4036.3	29.55	29.325	29.4
6	12	1	3	14	3553.9	30.65	30.575	30.6
7	12	0	2	16	1986.7	30.65	29.775	30.067
8	1	11	2	16	20924	11.9	12.025	11.983
9	1	12	2	16	22936	17.85	12.7	14.417
10	1	11	1	18	24191	11.65	14.475	13.533
11	15	1	2	16	3993.3	27.2	27.225	27.217
12	16	0	2	16	1933.8	30.35	30.35	30.35
13	1	12	2	16	22953	13.2	13.125	13.15
14	1	11	2	16	21065	10.95	12.475	11.967
15	1	12	2	16	23517	23.1	14.725	17.517
16	1	11	2	16	21040	11.25	12.55	12.117
17	1	10	2	16	19167	12.15	12.5	12.383
18	1	11	2	16	21094	12.1	14.375	13.617
19	1	10	2	16	19026	18.15	11.775	13.9
20	1	10	2	16	19209	14.3	14.3	14.3
21	1	7	2	16	14552	14.25	14.4	14.35
22	1	8	2	16	16081	12.7	12.725	12.717
23	2	7	2	16	14206	14.95	11.6	12.717
24	3	6	2	16	12812	15.5	14	14.5
25	1	9	2	16	17981	24.05	13.5	17.017
26	3	4	1	18	10925	15.65	15.475	15.533
27	2	5	1	18	12658	15.95	16.1	16.05
28	1	8	3	15	15323	11.15	12.8	12.25
29	1	12	2	16	23102	10.2	12.825	11.95
30	2	6	3	15	12080	16.2	12.9	14
31	2	6	2	16	12702	17.35	13.525	14.8
32	**1**	**11**	**2**	**16**	**21054**	**9.05**	**12.45**	**11.317**
33	2	10	2	16	19240	12.2	12.025	12.083
34	**1**	**12**	**2**	**16**	**23006**	**9.75**	**11.775**	**11.1**

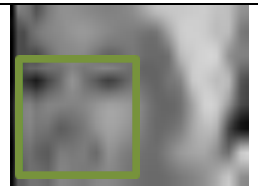
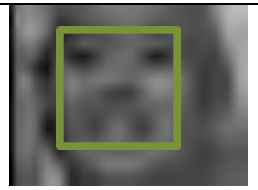
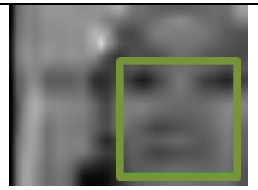
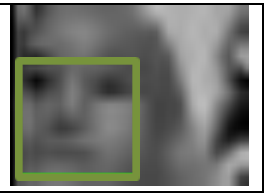
No. Fitur	posisi baris	panjang baris	posisi kolom	panjang kolom	threshold	true negative	false positive	sumError
35	1	11	2	16	21067	9.9	12.875	11.883
36	1	12	2	16	23074	9.45	13.625	12.233
37	1	18	1	18	41507	14.4	13.95	14.1
38	1	18	1	18	40824	16.95	21.075	19.7
39	1	18	1	18	40522	20.8	19.225	19.75

LAMPIRAN D

SKENARIO POSISI WAJAH FRONTAL DAN MIRING

No.					
1					
2					
3					
4					

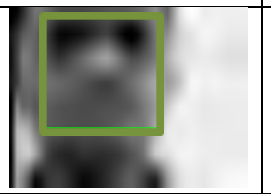
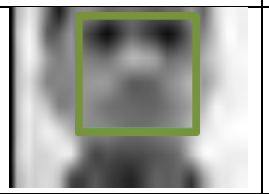
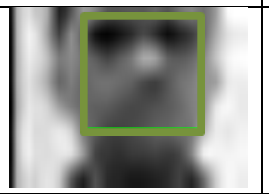
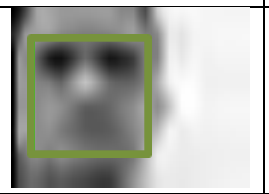
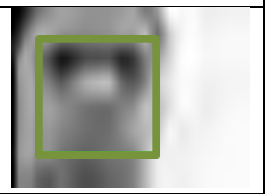
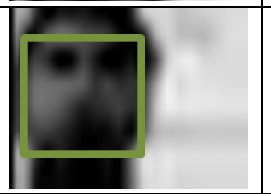
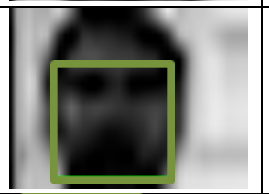
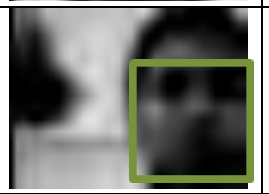
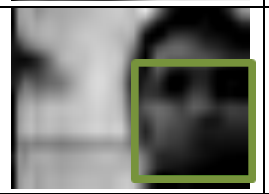
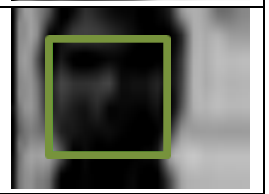
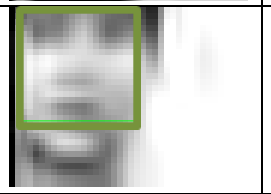
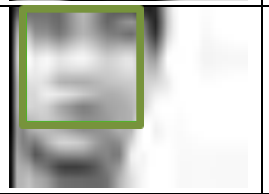
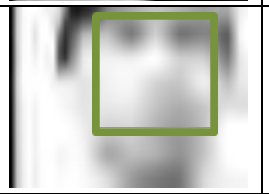
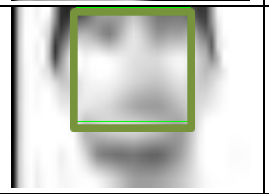
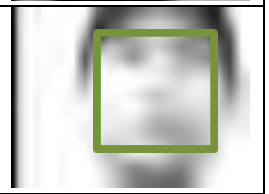
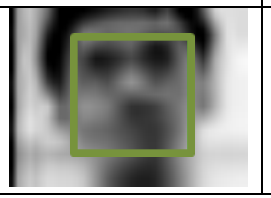
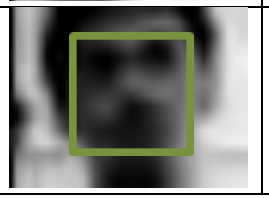
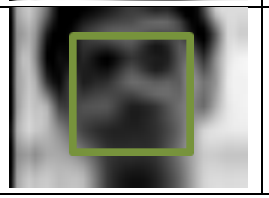
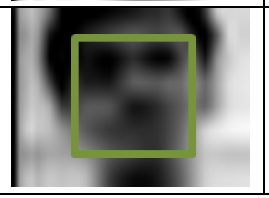
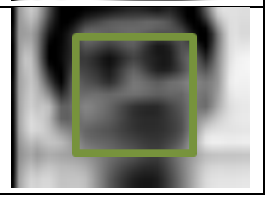

5
6
7
8
9

10					

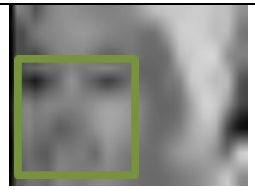
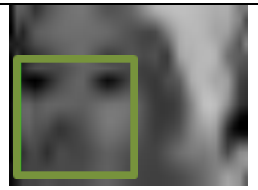
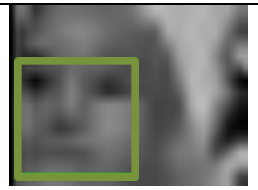
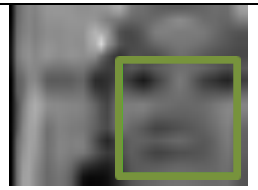
Halaman ini sengaja dikosongkan

LAMPIRAN E

SKENARIO MENDETEKSI KOORDINAT WAJAH

No.					
1					
2					
3					
4					

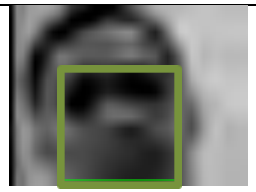
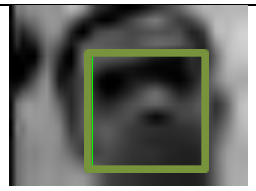
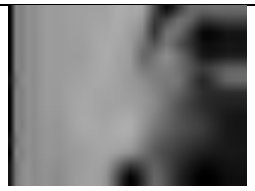
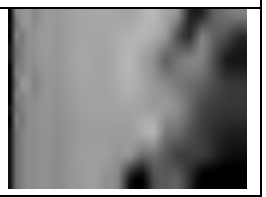
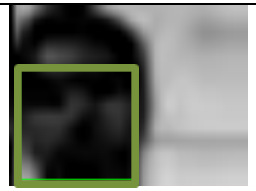
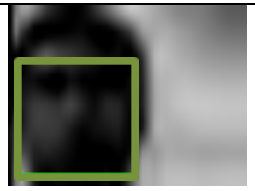
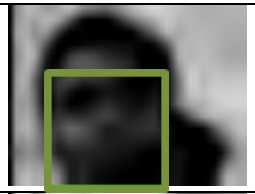
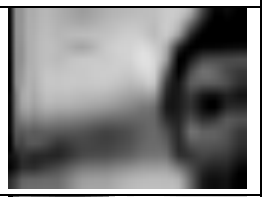
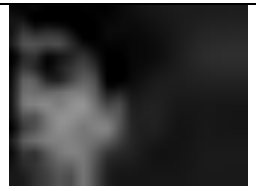
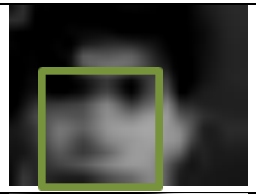
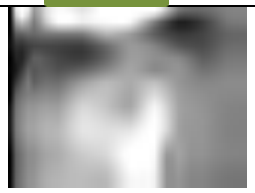
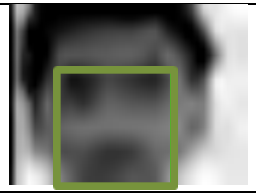
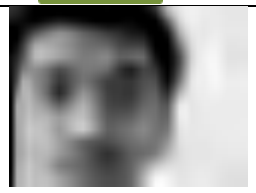
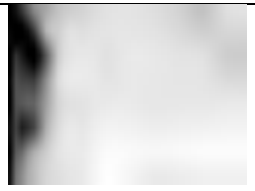
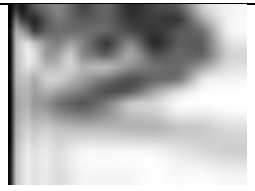
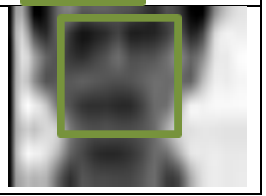

5
6
7
8
9

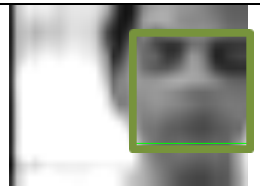
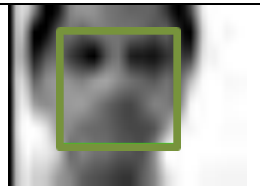
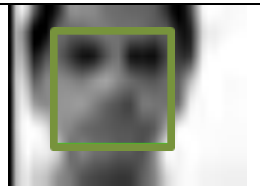
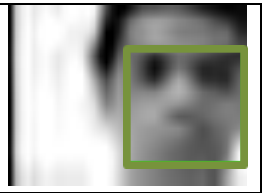
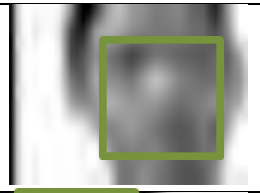
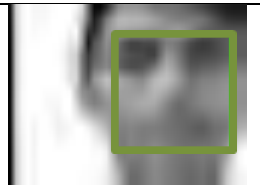
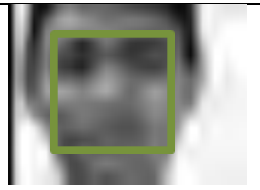
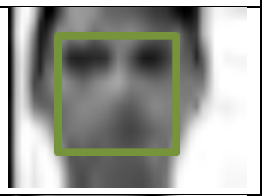
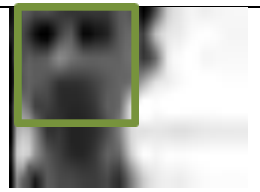
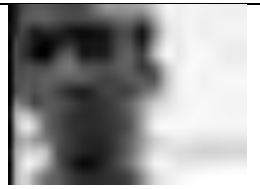
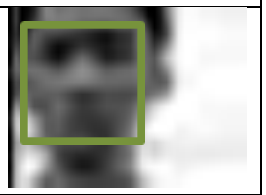
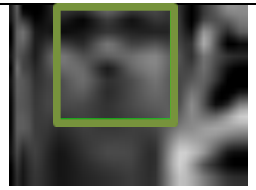
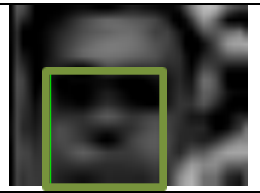
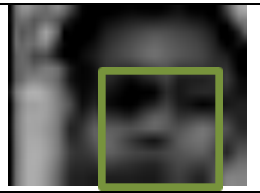
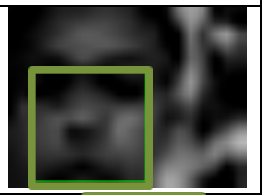
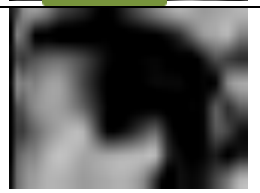
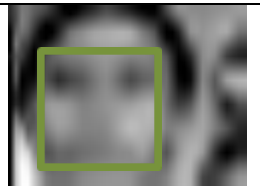
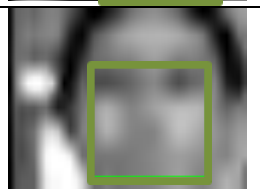
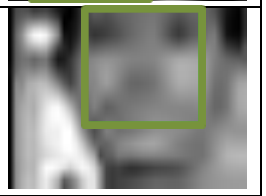
10					

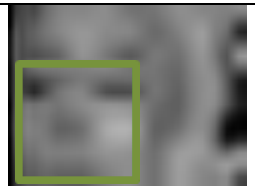
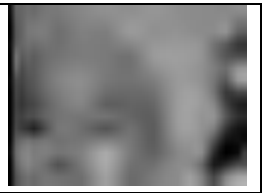
Halaman ini sengaja dikosongkan

LAMPIRAN F

SKENARIO WAJAH YANG BERGERAK

No.					
1					
2					
3					
4					

5					
6					
7					
8					
9					

10					

Halaman ini sengaja dikosongkan