TESIS

STUDI EVALUASI KINERJA BANANA PI M3 DALAM MENANGANI KOMPUTASI PARALEL MENGGUNAKAN OPENMP PADA APLIKASI PENGENALAN WAJAH

Disusun dan diajukan oleh

ASRARUL IKRAM
D032171020

TEKNIK ELEKTRO
FAKULTAS TEKNIK SEKOLAH PASCASARJANA
UNIVERSITAS HASANUDDIN
MAKASSAR
2021

STUDI EVALUASI KINERJA BANANA PI M3 DALAM MENANGANI KOMPUTASI PARALEL MENGGUNAKAN OPENMP PADA APLIKASI PENGENALAN WAJAH

Tesis

Sebagai Salah Satu Syarat Untuk Mencapai Gelar Magister

Program Studi

Teknik Elektro

Disusun dan diajukan oleh

ASRARUL IKRAM

Kepada

PROGRAM PASCASARJANA
UNIVERSITAS HASANUDDIN
MAKASSAR
2021

LEMBAR PENGESAHAN TESIS

STUDI EVALUASI KINERJA BANANA PI M3 DALAM MENANGANI KOMPUTASI PARALEL MENGGUNAKAN OPENMP PADA APLIKASI PENGENALAN WAJAH

Disusun dan diajukan oleh

ASRARUL IKRAM

D032171020

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Magister Program Studi Teknik Elektro Fakultas Teknik Universitas Hasanuddin pada tanggal 13 Oktober dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
Adnan, S.T., M.T., Ph.D	**Prof. Dr. Eng. Syafaruddin, S.T., M.Eng.**
Nip. 197404262003121002	Nip. 197405301999031003

Ketua Program Studi,	Dekan Fakultas Teknik,
Prof. Dr. Eng. Syafaruddin, S.T.,M.Eng.	**Prof. Dr. Ir. Muh. Arsyad Thaha, M.T.**
Nip. 197405301999031003	Nip. 196012311986091001

PERNYATAAN KEASLIAN TESIS

Yang bertanda tangan di bawah ini:

Nama	: ASRARUL IKRAM
Nomor Pokok	: D032171020
Program Studi	: Teknik Elektro
Konstentrasi	: Teknik Informatika

Menyatakan dengan sebenarnya bahwa tesis yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilan tulisan atau pemikiran orang lain. Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan tesis ini hasil karya orang lain, saya bersedia menerima sanksi atas perbuatan tersebut.

Gowa, 27 Oktober 2021

Yang menyatakan,

ASRARUL IKRAM

KATA PENGANTAR

Alhamdulillah, segala puji dan syukur kepada Allah SWT yang Maha Kuasa yang telah memberikan rahmat, hidayah dan pertolongannya kepada kami dalam menyelesaikan hasil penelitian ini, yang berjudul "**STUDI EVALUASI KINERJA BANANA PI M3 DALAM MENANGANI KOMPUTASI PARALEL MENGGUNAKAN OPENMP PADA APLIKASI PENGENALAN WAJAH**".

Ucapan terima kasih penulis juga hantarkan kepada Pembimbing tesis, Bapak Adnan, S.T., M.T., Ph.D. dan Bapak Prof. Dr. Eng. Syafaruddin, ST. M.Eng. yang telah meluangkan waktunya kepada penulis untuk membimbing dan berkonsultasi serta meminta saran tentang materi dalam hasil penelitian ini dan juga kepada seluruh dosen dan staf Department Teknik Elektro, Universitas Hasanuddin yang telah membantu dalam hal keilmuan maupun administrasi pada tahap penyelesaian hasil penelitian tesis ini. Penulis mengucapkan terima kasih kepada istri Fatmawati atas dukungan semangat yang tiada henti-hentinya. Dan juga penulis mengucapkan terima kasih kepada kedua orangtua, Syamsurijal dan Rustati rahimahullah atas dukungan doa yang sangat berarti buat hidup ini. Serta kepada seluruh teman-teman S2 Teknik Informatika angkatan 2017 yang telah membantu dalam penyelesaian hasil penelitian ini.

Penulis menyadari bahwa hasil penelitian ini masih belum sempurna. Penulis memohon maaf jika masih terdapat banyak kekurangan. Dengan demikian, penulis tetap mengharapkan saran dan kritik dari para pembaca sekalian serta tetap mengharapkan semoga tulisan ini bisa memberikan manfaat kepada seluruh pihak.

Gowa, 27 Oktober 2021

Asrarul Ikram

ABSTRAK

Asrarul Ikram. Studi Terhadap Evaluasi Kinerja Banana Pi M3 Untuk Menangani Komputasi Paralel Menggunakan OpenMP Pada Aplikasi Pengenalan Wajah (dibimbing oleh Adnan dan Syafaruddin)

Kemunculan Banana Pi M3 menarik perhatian para peneliti dengan spesifikasi *octa core* yang dimilikinya. Penelitian ini bertujuan untuk mengukur bagaimana kemampuan papan tunggal Banana Pi M3 dalam menangani proses komputasi paralel dengan mengoptimalkan kinerja procesor. Banana Pi M3 mampu menjalankan program pengenalan wajah dengan algoritma Eigenface menggunakan OpenMP dan meningkatkan waktu eksekusi di beberapa bagian pada algoritmanya. Diantaranya adalah *covariance matrix*, menghitung *eigenface* dan menghitung bobot. Pengenalan wajah dilakukan pada sejumlah ukuran gambar yang berbeda, yaitu 92x112, 296x360, 394x480 dan 493x600. Metode yang kami gunakan adalah menambahkan direktif OpenMP pada algoritma dengan *Data Level Parallelism* menggunakan klaus *parallel task*. Dengan akurasi pengenalan wajah yang tidak berubah, dilakukan proses komputasi paralel. Hasil yang diperolah menunjukkan bahwa perangkat Banana Pi M3 mampu menangani proses komputasi paralel dengan baik dengan speedup terbaik menggunakan 8 thread yaitu pada gambar berukuran 92x112 mencapai 1,64 kali, gambar berukuran 296x360 mencapai 2,68 kali, gambar berukuran 394x480 mencapai 2,86 kali dan gambar berukuran 493x600 mencapai 2,89 kali.

Kata Kunci: **Banana Pi M3, Komputasi Paralel, OpenMP, *Task Level Parallelism*, *Eigenface***

ABSTRACT

Asrarul Ikram. Study on Performance Evaluation of Banana Pi M3 to Handle Parallel Computing Using OpenMP in Face Recognition Application (supervised by Adnan and Syafaruddin)

The appearance of the Banana Pi M3 cough our attention with its octa-core specifications. This paper aims to measure how a single board Banana Pi M3 can handle parallel computing processes by optimizing processor performance. Banana Pi M3 is capable of performing face recognition programs with the Eigenface algorithm using OpenMP and improve the execution time of a number of algorithms. Among them are covariance matrices, calculate eigenfaces and calculate weights. Face recognition is performed on a number of different image sizes, namely 92x112, 296x360, 394x480 and 493x600. Our proposed method is to add an OpenMP directive to the algorithm with Data Level Parallelism using a parallel task clause. With facial recognition accuracy that does not change, a parallel computing process is carried out. The results obtained show that the Banana Pi M3 device is able to handle parallel computing processes well with the best speedup using 8 threads, namely the image size of 92x112 reaches 1.64 times, the image size of 296x360 reaches 2.68 times, the image size of 394x480 reaches 2.86 times. and the image size is 493x600 reaching 2.89 times.

Keyword: Banana Pi M3, Parallel Computing, OpenMP, Task Level Parallelism, Eigenface

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

A. Latar Belakang

Seiring berjalannya waktu di era saat ini, teknologi sudah berkembang sedemikian cepatnya dan telah membawa dunia memasuki era yang baru. Ketika mainframe komputasi sudah membutuhkan pemeliharaan yang tinggi dan sulit untuk dapat diakses, personal komputer ditemukan. Ketika bahasa pemrograman struktural tidak mampu menyediakan abstraksi yang dibutuhkan, pemrograman berorientasi objek ditemukan. Ketika high frequency scaling pada prosesor membutuhkan pendingin yang sangat mahal, multi core processor ditemukan (Govindaraj, 2016). Begitu juga saat ini ketika data input sudah semakin banyak dan ukuran datanya sudah semakin besar, teknologi baru juga dimunculkan seperti Big Data, Internet of Things, Machine Learning, Deep Learning, Artificial Intelegence, dan Virtual Reality (Suhendra et al., 2018).

Teknologi tersebut memiliki proses komputasi yang sangat besar dan kompleks sehingga berbagai macam perangkat yang baru pun telah bermunculan untuk menanganinya (Govindaraj, 2016). Perangkat pengolah data dapat dibangun menjadi komputer super untuk menangani proses komputasi yang sangat besar. Sehingga konsumsi daya yang dibutuhkan sangat besar dan cukup mahal (Yang et al., 2010).

Sejauh ini data input sanggup dikomputasi dengan pemrograman sekuensial karena ukurannya terbilang kecil. Pada dasarnya proses komputasi juga dapat dilakukan walaupun ukuran data inputnya besar, tetapi akan membutuhkan waktu proses yang lama. Maka dari itu pemrograman paralel dibutuhkan untuk mempercepat proses komputasi (Matthews et al., 2018). Dengan pemrograman paralel, perlunya peneliti memahami kompleksitas dan kasus-kasus terkait dengan memparalelkan data.

Citra merupakan salah satu input data yang memiliki ukuran yang besar. Dan saat ini sudah banyak penelitian tentang pengolahan citra seperti pengenalan wajah yang menjadi daya tarik tersendiri dalam penelitian. Ada studi ekstensif pengenalan wajah dalam beberapa tahun terakhir dengan relevansi berbagai domain dalam meningkatkan akurasi seperti dua dimensi, tiga dimensi, dan video yang telah memberikan kontribusi besar pada bidang penelitian dan pengembangan yang sama (Kurniawan et al., 2017; Zafaruddin & Fadewar, 2018). Semakin besar input data maka semakin lama proses komputasinya, sehingga menarik perhatian kami untuk melakukan penelitian ini.

Dilihat dari ukuran data input, teknik komputasi seperti pemrograman paralel sangat dibutuhkan pada pekerjaan ini. Penelitian seputar pemrograman paralel juga sudah banyak dilakukan seperti pemrograman paralel hibrid pada klaster GPU dengan merancang model algoritma menggunakan bahasa pemrograman CUDA dari NVIDIA yang berjalan

secara *multi-thread* dan *multi-core* GPU. Teknik *Message Passing Interface* (MPI) juga dilakukan untuk mengklaster GPU. Hal yang sama juga sudah dilakukan pada perangkat yang tergolong murah dan hemat energi seperti klaster raspberry pi (Govindaraj, 2016; Yang et al., 2010). Pemrograman paralel juga sudah berhasil dilakukan pada algoritma *Decision Tree* dan C4.5 dengan penyederhanaan transformasi algoritma sekuensial ke bentuk paralel (Kholod et al., 2017).

Untuk menjalankan pemrograman paralel perlunya dipahami sistem *multi-processor*, *multi-core* dan *multi-thread*. Sistem *multi-processor* berisi lebih dari satu CPU (*Central Processing Unit*) dan memungkinkan mereka untuk bekerja secara paralel. Dalam sistem *multi-processor* terdapat dua atau lebih pemrosesan CPU yang terintegrasi ke dalam satu sistem komputer. Jadi pada dasarnya ini memiliki CPU dua atau lebih pada sistem secara fisik. Secara bersamaan prosesor melakukan instruksi program yang berbeda sehingga mempercepat proses komputasi. Sistem *multi-core* terdapat dalam sebuah CPU yang memiliki lebih dari satu *core* dan dapat dijalankan secara paralel. Pada dasarnya arsitektur sistem *multi-core* diduplikasi dari sistem *multi-processor*, sehingga beberapa *core* dapat bekerja secara paralel pada operasi yang terpisah (Tabassum et al., 2016). Dan *multi-thread* merupakan salah satu kemampuan CPU yang setiap *core*-nya mampu mengeksekusi banyak proses secara bersamaan.

Program pengenalan wajah menggunakan algoritma eigenface telah dibuat sebelumnya (Adreeva, n.d.). Program dirancang menggunakan

bahasa C++ secara serial. Meskipun program ini sangat membantu dalam mengenali wajah, namun program tersebut kurang ideal karena memiliki waktu eksekusi yang lama.

(Lindner et al., 2020) juga telah menguji berbagai jenis komputer papan tunggal, salah satunya adalah Banana Pi M3. Program pengenalan wajah juga diuji pada perangkat Banana Pi M3, tetapi dilakukan secara serial, bukan paralel.

Melihat dari pengerjaan sebelumnya, kami juga menawarkan pengukuran dengan Banana Pi M3 sebagai perangkat yang akan digunakan untuk proses komputasi paralel. Memiliki spesifikasi yang lebih unggul yaitu delapan core processor menjadi daya tarik untuk melakukan penelitian terhadap evaluasi kinerja pada perangkat Banana Pi M3.

Untuk pengujian dan analisis pada penelitian ini, proses komputasi dalam pengenalan wajah dipilih sebagai target implementasi. Proses komputasi pengenalan wajah secara sekuensial dikerjakan pada perangkat Banana Pi M3 memiliki estimasi waktu yang sangat lama. Maka diharapkan dengan adanya pemrograman paralel menggunakan direktif OpenMP, proses komputasi dapat berjalan lebih cepat. Perhitungan speedup dilakukan dengan du acara yaitu dengan yang sebenarnya dan menggunakan hukum Amdahl. Dengan adanya kesenjangan diantara kedua speedup, analisis dilakukan untuk mencari waktu overhead yang terjadi.

B. Rumusan Masalah

Berdasarkan latar belakang yang ada, dapat dirumuskan masalah sebagai berikut:

1. Bagaimana kinerja perangkat Banana Pi M3 dalam menangani proses komputasi paralel?
2. Bagaimana kinerja OpenMP terhadap program pengenalan wajah dengan algoritma eigenface?
3. Bagaimana pengaruh overhead terhadap komputasi paralel pada perangkat Banana Pi M3?

C. Tujuan Penelitian

Adapun tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Untuk mengetahui kinerja perangkat Banana Pi M3 dalam menangani proses komputasi paralel.
2. Untuk mengetahui kinerja OpenMP terhadap program pengenalan wajah dengan algoritma eigenface.
3. Untuk mengetahui pengaruh overhead terhadap komputasi paralel pada perangkat Banana Pi M3.

D. Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Kepada Akademisi: memberikan informasi tentang pengukuran komputasi paralel pada perangkat Banana Pi M3 yang dapat dijadikan referensi dalam menyelesaikan penelitian-penelitian kedepannya.

2. Kepada Industri: menjadi *feedback* agar kedepannya membangun perangkat yang lebih baik.
3. Kepada Masyarakat: menjadi literatur atau memberitahukan tentang adanya perangkat baru yang lebih hemat energi dan lebih murah.

E. Batasan Masalah

Adapun batasan-batasan dalam penelitian ini adalah:

1. Penelitian ini hanya membahas pada kinerja perangkat Banana Pi M3 dalam menangani proses komputasi pengenalan wajah menggunakan pemrograman paralel, tidak mengkaji masalah komunikasi datanya.
2. Perangkat yang digunakan untuk pengolahan citra adalah perangkat Banana Pi M3.
3. *Directive* yang digunakan dalam pemrograman paralel adalah OpenMP.
4. Citra wajah yang diproses merupakan citra diam berukuran 92x112, 296x360, 394x480 dan 493x600 dengan format pgm.

BAB II

TINJAUAN PUSTAKA

A. Komputasi Paralel

Komputer paralel adalah komputer dengan multi prosesor, atau prosesor dengan banyak inti yang bekerja sama untuk menyelesaikan masalah dengan melakukan tugas secara bersamaan pada bagian yang berbeda. Ini memungkinkan komputer menjalankan beberapa instruksi sekaligus. Akibatnya, waktu konsumsi untuk menyelesaikan masalah berkurang secara signifikan (Suhendra et al., 2018).

Komputasi paralel adalah suatu bentuk komputasi dimana banyak perhitungan dilakukan secara bersamaan, beroperasi berdasarkan prinsip bahwa masalah besar sering kali dapat dibagi menjadi masalah yang lebih kecil, yang kemudian diselesaikan secara bersamaan. Terdapat beberapa tingkatan komputasi paralel yang seperti level bit, level instruksi, data, dan paralelisme tugas (Mustafa et al., 2015). Beberapa teknik untuk melakukan komputasi paralel yang paling terkenal menggunakan *Message Parsing Interface* (MPI), *Parallel Virtual Machine* (PVM), dan OpenMP.

Saat ini permintaan komputasi paralel sudah sangat luas, dan ada tiga jenis permintaan yang utama(Li & Zhang, 2019) :

1. Aplikasi komputasi intensif, seperti kalkulasi rekayasa ilmiah skala besar dan simulasi numerik.

2. Aplikasi data intensif, seperti perpustakaan digital, *data warehouse*, *data mining*, visualisasi komputasi.
3. Aplikasi jaringan intensif, seperti kerja sistem, kendali jarak jauh, dan diagnosis medis jarak jauh.

B. OpenMP

OpenMP (*Open Multi Processing*) adalah sebuah API (*Application Programming Inteface*) yang mendukung pemrograman dalam bahasa pemrograman C, C++, dan FORTRAN. OpenMP diusulkan dan dirancang untuk mesin dengan arsitektur *multi-processor*, *multi-core*, dan *shared-memory*. OpenMP mendefinisikan model portabel dan skalabel dengan antarmuka yang sederhana dan fleksibel untuk mengembangkan aplikasi paralel pada platform dari desktop hingga supercomputer (*Home - OpenMP*, n.d.).

OpenMP menyediakan abstraksi atas detail-detail mengenai cara eksekusi *thread* di level rendah. OpenMP mendukung paralelisme bertingkat dengan kemampuan setiap *thread* yang berjalan untuk membuat *thread-thread* turunannya sendiri. Jika tidak digunakan dengan benar paralelisme bertingkat dapat menyebabkan kelebihan permintaan prosesor, yang dapat menyebabkan penurunan kinerja yang signifikan (Mathews & Abraham, 2017).

Shared-memory berarti memori utama yang dapat dibagikan ke sejumlah N prosesor. Proses komunikasi antara dua atau lebih prosesor dapat dilakukan melalui *shared-memory*. Semua akses *thread* ke *shared-*

memory bersifat global. Ada dua jenis data yang dapat digunakan yaitu *shared* dan *private*. Data *shared* dapat diakses oleh semua *thread* sedangkan data *private* hanya dapat diakses oleh *thread* yang memilikinya (Suhendra et al., 2018).

Model eksekusi paralel yang diterapkan oleh OpenMP yaitu model *fork-join*. Dalam model ini terdapat dua *thread*. Pertama adalah *thread* utama yang menjalankan proses tunggal yang merupakan proses awal saat eksekusi dimulai. Ini berjalan secara berturut-turut sebelum bagian paralel pertama ditemukan. Selanjutnya sebelum memasuki bagian paralel *thread* kedua dibuat dan dinamakan grup *thread* paralel. Baris kode yang terdapat pada bagian paralel dijalankan secara paralel di antara grup *thread*. Setelah *thread* ini menyelesaikan tugas-tugasnya proses sinkronisasi dilakukan kemudian dihentikan dan hanya *thread* utama yang tersisa.

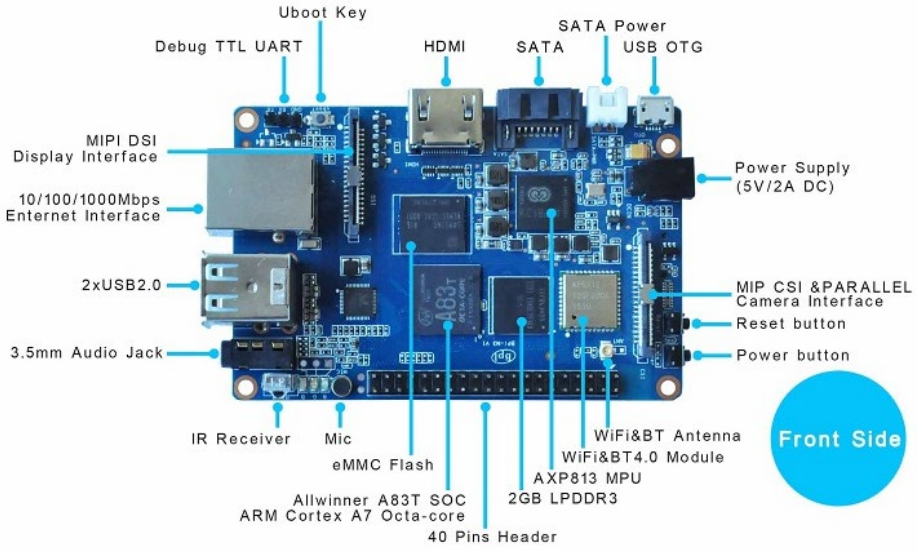
Untuk menentukan bagian paralel pada aplikasi dibutuhkan kode pragma yang dituliskan pada awal pengkodean paralel. #pragma omp parallel untuk C/C++ dan !$OMP PARALLEL untuk Fortran. Untuk menggunakan kode ini perlu mendefinisikan OpenMP pada *header* aplikasi seperti #include <omp.h> pada C/C++. Selanjutnya di proses kompilasi perlu ditambahkan atribut -fopenmp untuk mengeksekusi perintah paralel yang terdapat pada aplikasi.

OpenMP telah digunakan dengan sukses untuk memparalel kan sejumlah besar aplikasi ilmiah di berbagai bidang. Sangat sering

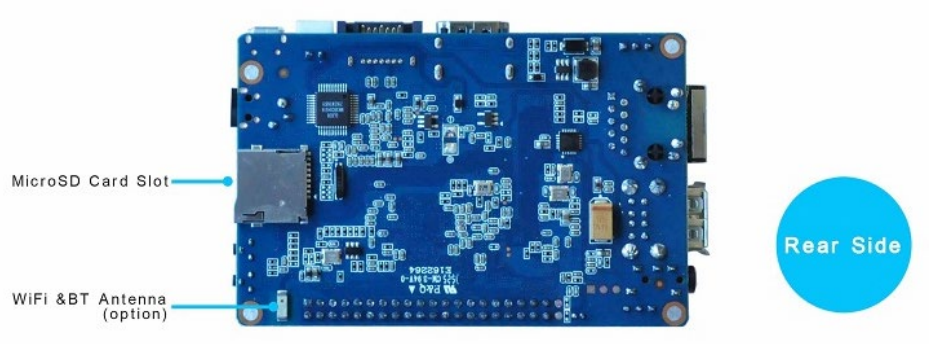
digunakan untuk memparalelkan aplikasi numerik. Banyak metode distribusi pekerjaan yang secara khusus dikembangkan untuk mendukung paralelisme pada proses perulangan dan rekursif (Mustafa et al., 2015).

C. Arsitektur Banana Pi M3

Banana Pi M3 adalah komputer *single board* yang dibebankan dengan prosesor ARM Cortex-A7 *octa-core* 1.8 GHz, memori RAM 2GB dan penyimpanan internal eMMC 8GB. Disamping itu, Banana Pi M3 juga memiliki *Gigabit Ethernet*, 2USB, slot microSD, SATA, WiFi, Bluetooth dan koneksi HDMI. Berbagai sistem operasi dapat berjalan pada Banana Pi M3 seperti Android versi 5.1.1 dan beberapa distro Linux Ubuntu, Debian, Armbian, Rasbian, Arch, FreeBSD, Simplenas, LakkaTV dan lain-lain (*Banana Pi BPI-M3 - Banana Pi Wiki*, n.d.). Tampak atas pada perangkat Banana Pi M3 dapat dilihat pada Gambar 2.1. Dan tampak bawah pada perangkat Banana Pi M3 dapat dilihat pada Gambar 2.2.



Gambar 2. 1 Banana Pi M3 Tampak Atas



Gambar 2. 2 Banana Pi M3 Tampak Bawah

D. Pengenalan Wajah

Wajah adalah bagian penting dari tubuh yang berperan penting dalam identifikasi manusia. Teknologi yang memetakan fitur wajah secara matematis dan menyimpan data yang membantu dalam memverifikasi atau mengidentifikasi seseorang secara unik dapat disebut sebagai Pengenalan Wajah. Teknologi pengenalan wajah memiliki karakteristik perolehan yang mudah dan keandalan yang tinggi yang telah banyak digunakan di berbagai bidang seperti sistem keamanan, penyelidikan forensik, pemantauan lalu lintas, aplikasi seluler dan lain-lain (Qu et al., 2018).

Pengenalan Wajah dapat dilakukan dengan dua pendekatan yaitu Metode Holistik dan Metode berbasis Fitur Local. Dalam metode Holistik, sistem pengenalan menganggap seluruh citra wajah sebagai masukan. Dalam metode berbasis Fitur Lokal, fitur lokal seperti mata, hidung, bibir, dan lain-lainl diekstraksi dan statistik lokalnya digunakan sebagai masukan untuk pengenalan wajah. PCA adalah algoritma Holistik yang terkenal untuk melakukan pengenalan wajah menggunakan konsep Eigenfaces (Sapna et al., 2019).

E. Algoritma Eigenface

Kata “eigenface” berasal dari kata Jerman “eigenwert”. Kata “eigen” mengacu pada karakteristik dan kata “wert” mengacu pada nilai. Eigenface adalah algoritma pengenalan wajah berdasarkan *Principal Component Analysis* (PCA) yang dikembangkan oleh Massachusetts Institute of Technology (MIT). Eigenface digunakan untuk mereduksi dimensionalitas dan mencari vektor terbaik untuk mendistribusikan citra (Kurniawan et al., 2017; Wahyu Mulyono et al., 2019). Banyak peneliti yang lebih suka menyebutnya dengan *eigen image*. Teknik ini telah digunakan dalam pengenalan tulisan tangan, membaca libs, pengenalan suara dan pencitraan medis (Putranto et al., 2017).

Berikut adalah Langkah-langkah algoritma eigenface:

1. Ambil semua kelas dataset wajah lalu direpresentasikan sebagai F_1, F_2, F_3, F_4, … F_N dengan ketentuan setiap citra memiliki wajah yang berada di tengah citra dan setiap citra memiliki ukuran piksel yang sama (Sapna et al., 2019).
2. Setiap data F_N ditransformasi menjadi matriks $M = 1 \times F_{weight} \times F_{height}$. Maka didapatkan data latih berupa matrik pada vektor $A_i = n \times F_{weight} \times F_{height}$ seperti pada persamaan 1 (Kurniawan et al., 2017; Putranto et al., 2017).

$$A = [F_1, F_2, F_3, \dots, F_N] \qquad (1)$$

3. Hitung rata-rata vektor. Pada persamaan 2 (Kurniawan et al., 2017; Putranto et al., 2017), setelah set data latih siap, nilai rata-rata di

setiap piksel dalam vektor perlu dihitung untuk mendapatkan citra rata-rata.

$$B = \frac{1}{N}\sum_{i=1}^{N} A_i \qquad (2)$$

4. Kurangi data latih terhadap citra rata-rata seperti pada persamaan 3 (Kurniawan et al., 2017; Putranto et al., 2017). Semua gambar rata-rata dikurangi kemudian digabungkan bersama untuk mendapatkan matriks pengurangan rata-rata akhir.

$$A_i = A_i - B \qquad (3)$$

5. Hitung nilai matrik kovarian. Dilihat dari persamaan 4 (Putranto et al., 2017), matriks kovarian dapat dihitung dengan mengalikan data latih yang telah dikurangi dengan citra rata-rata dan transposnya. Ukuran matriks kovariansi akan sama dengan *N* × *N* dimana *N* adalah jumlah gambar pada set data latih (Sapna et al., 2019).

$$S = A \times A^T \qquad (4)$$

6. Hitung nilai eigenvector dan eigenvalue dari matriks kovarian. Langkah-langkah untuk menemukan eigenvector dan eigenvalue diadopsi dari metode Jacobi, yaitu dengan melakukan rotasi data terus-menerus hingga nilai elemen matriks selain diagonal menjadi 0. Selanjutnya nilai dalam matriks diurutkan berdasarkan eigenvalue terbesar sehingga mendapatkan nilai pada element diagonal matriks yang akan menjadi eigenvector (Borkar & Kuwelkar, 2018).

7. Menghitung eigenface dengan mengalikan matriks eigenvector dan matriks yang telah dikurangi dengan citra rata-rata terlihat pada persamaan 5. Dengan sampel berjumlah *E* eingenface berkisar dari 1 sampai *E*. Pencarian jumlah eigenface dilakukan untuk mendapatkan tingkat akurasi yang tinggi (Borkar & Kuwelkar, 2018).

$$U_e = \sum_{j=1}^{E} V_j \times A_j \qquad (5)$$

8. Menghitung bobot dengan mengalikan eigenface dan transpos matriks yang telah dikurangi citra rata-rata terlihat pada persamaan 6. Nilai bobot ditransformasi ke dalam matrik *E* × *N*, sehingga dapat dilihat piksel-piksel sampel wajah memiliki nilai bobot masing-masing di setiap eigenfacenya (Borkar & Kuwelkar, 2018).

$$W_e = \sum_{k=1}^{N} U_k \times A_k^T \qquad (6)$$

9. Tahap pengenalan wajah dengan mengubah wajah menjadi vektor matriks kemudian dikurangi dengan citra rata-rata. Menghitung bobot citra dengan mengalikan matriks eigenface dan transpos citra wajah yang ingin dikenali. Selanjutnya menghitung jarak antara bobot yang baru saja didapatkan dengan bobot yang sudah lama didapatkan dengan *Euclidean Distance*. Proses yang sama dilakukan di setiap wajah orang menggunakan foto yang beda untuk menghitung tingkat akurasinya (Kurniawan et al., 2017).

F. Metode Jacobi

Wajah bersifat multidimensi dan oleh karena itu memiliki "*Curse Of Dimensionality*" yaitu wajah membutuhkan banyak memori dan waktu untuk diproses. Untuk mengatasi masalah ini, fitur optimal harus diperoleh untuk meningkatkan akurasi dan menghilangkan noise dari gambar. PCA banyak digunakan untuk mengurangi dimensionalitas.

Selanjutnya setelah mengurangi dimensi, gambar diproyeksikan ke ruang *eigen* menggunakan LDA. Untuk melakukan hal tersebut diperlukan perhitungan *eigenvalues* dan *eigenvectors*. Banyak metode yang dapat digunakan untuk menghitung *eigenvalues* dan *eigenvectors* seperti metode QR, metode *Gauss-Seidel*, metode *Power*, metode *Jacobi*, dan lain-lain. Dalam penelitian ini digunakan metode jacobi karena metode *Jacobi* merupakan metode iteratif untuk mencari *eigenvalues* dan *eigenvectors* dari matriks simetris (Borkar & Kuwelkar, 2018).

G. Jarak Euclidean

Jarak *Euclidean* atau metrik *Euclidean* adalah jarak antara dua titik dalam ruang *Euclidean*. Ruang *Euclidean* diperkenalkan oleh Euclid, seorang matematikawan dari Yunani untuk mempelajari hubungan antara sudut dan jarak. Hal ini berkaitan dengan teorema *Pythagoras Euclidean* dan biasanya diterapkan pada berbagai dimensi di bidang geometri (Kurniawan et al., 2017).

Jarak *Euclidean* dihitung antara fitur gambar uji dan fitur gambar pelatihan. Citra wajah dengan ukuran *Euclidean* minimum dianggap cocok

untuk citra uji. Batas ditetapkan untuk jarak *Euclidean* yang dapat membantu dalam menentukan gambar yang tidak dikenali dari gambar yang dikenali. Gambar uji apa pun dengan jarak *Euclidean* minimum lebih besar dari batas yang ditetapkan dianggap tidak dikenali (Sapna et al., 2019).

H. Hukum Amdahl

Semakin cepat aplikasi berjalan, semakin sedikit waktu yang dibutuhkan untuk menunggu hasilnya. Waktu eksekusi yang lebih pendek memungkinkan pengguna untuk menjalankan komputasi dengan kumpulan data yang lebih besar (lebih banyak piksel dalam penelitian ini) dalam jumlah waktu yang dapat diterima. Untuk menunjukkan perbedaan waktu komputasi antara eksekusi serial dan paralel, perlu ada variabel yang dapat memberi tahu kita peningkatan kinerja saat menggunakan paralel dibanding serial. Salah satu variabel yang dapat menunjukkan perbandingan nyata antara eksekusi serial dan paralel adalah speedup.

Secara sederhana, speedup adalah rasio waktu eksekusi serial terhadap waktu eksekusi paralel. Dalam analisis penskalaan kinerja komputasi paralel, seseorang dapat menggunakan hukum Amdahl untuk menghitung batas atas pada percepatan aplikasi ketika kinerja paralel ditingkatkan (Che & Nguyen, 2014). Hukum Amdahl ditunjukkan pada persamaan 7. Dengan normalisasi waktu eksekusi serial menjadi 1, dan membaginya dengan perkiraan waktu eksekusi paralel yang menghasilkan speedup sebagai perolehan persentase waktu serial yang dinormalisasi,

diasumsikan sebagai *S*. Diasumsikan bahwa fraksi 1 – *B* adalah persentase eksekusi serial dan fraksi *B* adalah persentase eksekusi serial yang dapat dieksekusi secara paralel. Dan *N* adalah jumlah thread yang digunakan.

$$S = \frac{1}{(1-B) + \frac{B}{N}} \qquad (7)$$

Persamaan di atas benar selama tiga hal berikut berlaku. 1) Program yang dieksekusi tidak berubah dan bagian yang dapat diparalelkan juga tetap konstan; 2) Ada ruang memori yang tidak terbatas dan overhead tambahan untuk mengganti blok memori, blok disk, dll.; 3) overhead unit komputasi yang mencakup akses memori, komunikasi on-chip atau off-chip dan sinkronisasi inti dapat diabaikan (Pei et al., 2016).

I. Portable Gray Map

Portable Gray Map (PGM) adalah format gambar *grayscale* yang sangat jarang diperbincangkan. Format ini dirancang supaya mudah untuk dipelajari dan ditulis pada program. Sangat sederhana sehingga kebanyakan orang dapat merekayasa ulang filenya. File PGM terdiri dari urutan satu atau lebih gambar PGM. Tidak ada data, pembatas, atau *padding* sebelum, sesudah, atau diantara gambar. Setiap gambar PGM terdiri dari (Poskanzer, 1989):

1. Sebuah "*magic number*" untuk mengidentifikasikan jenis file. *Magic number* gambar PGM memiliki dua karakter "P5".
2. Lebar gambar, diformat sebagai karakter ASCII dalam decimal.

3. Tinggi gambar, diformat sebagai karakter ASCII dalam decimal.
4. Nilai maksimum abu-abu (MaxVal), diformat sebagai karakter ASCII dalam decimal. Harus kurang dari 65536 dan lebih dari nol.
5. Gambar yang diwakili dengan nilai abu-abu. Setiap baris terdiri dari nilai lebar gambar, berurutan dari kiri ke kanan. Setiap nilai abu-abu adalah angka dari 0 hingga MaxVal, dengan 0 berwarna hitam dan MaxVal berwarna putih. Setiap nilai abu-abu diwakili dalam biner murni oleh 1 atau 2 byte. Jika Maxval kurang dari 256, itu adalah 1 byte. Jika tidak, itu adalah 2 byte. Byte paling signifikan adalah yang pertama.
6. *String* yang dimulai dengan tanda "#" mungkin merupakan komentar.

Terdapat versi lain dari PGM yang sudah cukup langka yaitu format "*plain*" PGM. Format di atas yang umumnya dianggap normal dikenal sebagai format "*raw*" PGM. Perbedaannya dengan format *raw* adalah:

1. Hanya ada satu gambar dalam satu file.
2. *Magic number* bukan P5, tetapi P2.
3. Setiap piksel direpresentasikan sebagai angka desimal ASCII.
4. Setiap piksel memiliki spasi sebelum dan sesudahnya. Harus ada setidaknya satu karakter spasi di antara dua piksel, tetapi tidak ada nilai maksimum.
5. Tidak boleh ada baris yang lebih dari 70 karakter.

Berikut adalah contoh sebuah gambar kecil dalam format *plain* PGM.

```
P2
# eep.pgm
18 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

J. Penelitian Terkait

1. Olga Andreeva. 2014. ***Implementation of the Eigenface algorithm for face recognition in C++.*** *Source code* pada github.

   Review: Program pengenalan wajah dengan algoritma eigenface telah dibuat sebelumnya. Program dirancang menggunakan bahasa C++ secara serial. Meskipun program ini sangat membantu dalam mengenali wajah, namun program tersebut kurang ideal karena memiliki waktu eksekusi yang lama (Adreeva, n.d.).

2. Tymoteusz Lindner, Daniel Wyrwal, Marcin Bialek dan Patryk Nowak. 2020. ***Face recognition system based on a single-board computer***. *15th International Conference Mechatronic Systems and Materials, MSM* 2020.

   Review: Peneliti melakukan pengukuran berbagai jenis komputer papan tunggal, salah satunya adalah Banana Pi M3. Program pendeteksi wajah dan pengenalan wajah dijadikan sebagai target

implementasi pada penelitiannya. Algoritma yang digunakan dikerjakan secara serial, bukan paralel. (Lindner et al., 2020).

3. Sapna S, Anjali R, Shobha N Kamath. 2019. ***Performance Analysis of Parallel Implementation of PCA-based Face Recognition using OpenCL***. RTEICT - 2019 *4th IEEE International Conference on Recent Trends on Electronics, Information, Communication & Technologies*.

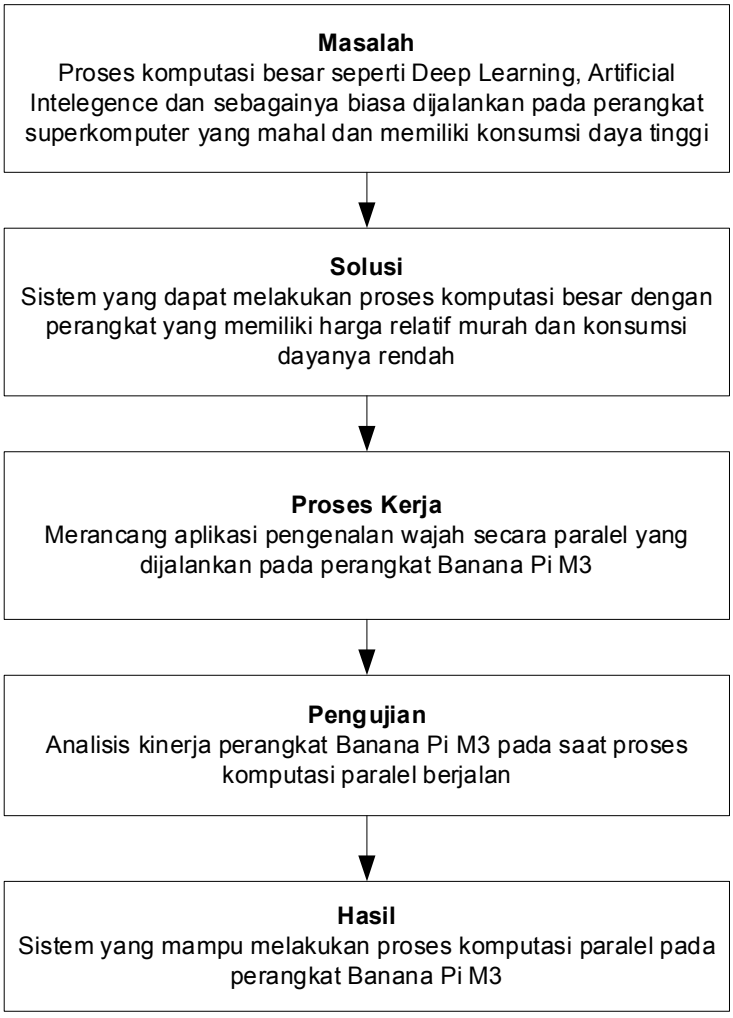
   Review: Pengenalan wajah sudah sering digunakan dalam berbagai bidang dan memerlukan proses komputasi yang cepat. Tetapi struktur wajah yang kompleks menyulitkan untuk mempercepat proses karena melibatkan perhitungan yang sangat besar. Algoritma PCA merupakan teknik sangat efektif dalam pengenalan wajah tetapi membutuhkan biaya komputasi yang lebih tinggi. Peneliti memparalelkan pengenalan wajah berbasis PCA menggunakan OpenCL pada Matlab untuk mempercepat komputasi dan dibuat perbandingan dalam hal waktu komputasi yang diperlukan untuk implementasi serial dan paralel. Eksekusi dilakukan pada prosesor grafis AMD Radeon. Paralel dilakukan pada empat tahap dalam algoritma PCA seperti: menghitung rata-rata, mengungari rata-rata, menghitung matriks kovarian dan menghitung *eigenvector*. Implementasi paralel menawarkan speedup 32 kali lipat dibandingkan implementasi serial yang membuktikan efisiensi metode yang diusulkan (Sapna et al., 2019).

4. Solmaz Salehian, Jiawen Liu, Yonghong Yan. 2017. ***Comparison of Threading Programming Models***. IPDPSW – 2017 IEEE International Parallel and Distributed Processing Symposium Workshops.

   Review Penelitian: Perbandingan fitur bahasa dan sistem runtime model pemrograman paralel yang umum digunakan untuk komputasi kinerja tinggi termasuk OpenMP, Intel Cilk Plus, Intel TBB, OpenACC, Nvidia CUDA, OpenCL, C++11 dan PThreads. Perbandingan kinerja antara OpenMP, Cilk Plus dan C++11 dilakukan untuk data dan paralelisme tugas pada CPU menggunakan berbagai macam aplikasi *banchmark*. Hasilnya menunjukkan bahwa kinerja bervariasi sehubungan dengan faktor-faktor seperti strategi penjadwalan, *overhead* terjadi dengan paralelisme dan sinkronisasi, penyeimbangan beban dan keseragaman beban kerja tugas di antara *thread* dalam aplikasi. *Overhead* yang tinggi terjadi pada program aplikasi paralel data karena serialisasi distribusi potongan loop di antara *thread* paralel. Tetapi, jika aplikasi memiliki penyeimbangan beban yang memadai dan beban kerja tugas yang efisien dengan penjadwalan dinamis, efek biaya *overhead* yang dibuat oleh waktu proses bisa jadi lebih kecil (Salehian et al., 2017).

K. Kerangka Pikir

Adapun kerangka piker dapat dilihat pada Gambar 2.3



Gambar 2. 3 Kerangka Pikir Penelitian