PERBANDINGAN TINGKAT AKURASI METODE *SUM OF ABSOLUTE DIFFERENCES* (SAD) DAN *SUM OF SQUARED DIFFERENCES* (SSD) DALAM IDENTIFIKASI POLA AKSARA LAMPUNG

(Skripsi)

Oleh:

ANISA RAHMADINI

NPM 2015031019

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

2025

ABSTRAK

PERBANDINGAN TINGKAT AKURASI METODE *SUM OF ABSOLUTE DIFFERENCES* (SAD) DAN *SUM OF SQUARED DIFFERENCES* (SSD) DALAM IDENTIFIKASI POLA AKSARA LAMPUNG

Oleh

ANISA RAHMADINI

Aksara Lampung merupakan salah satu warisan budaya yang memiliki nilai historis dan identitas bagi masyarakat Lampung. Namun, di era digital, aksara ini semakin jarang digunakan, sehingga diperlukan upaya pelestarian dengan memanfaatkan teknologi pengenalan pola. Penelitian ini membandingkan metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dalam identifikasi pola aksara Lampung berdasarkan tingkat akurasi dan efisiensi waktu komputasi. Dataset yang digunakan terdiri dari 6.500 karakter aksara Lampung, yang diuji dalam tiga skenario jumlah dataset (1.300, 3.900, dan 6.500 karakter) serta dua ukuran citra (40×40 piksel dan 80×80 piksel). Hasil penelitian menunjukkan bahwa kedua metode memiliki tingkat akurasi yang hampir identik, dengan akurasi tertinggi sebesar 90,54% pada dataset terbesar dengan ukuran citra 80×80 piksel. Namun, metode SAD secara konsisten lebih cepat dibandingkan metode SSD dalam semua skenario pengujian. Pada dataset 6.500 karakter, metode SAD membutuhkan waktu rata-rata 0,005 detik, sedangkan metode SSD memerlukan 0,534 detik. Dengan demikian, metode SAD lebih direkomendasikan untuk implementasi sistem pengenalan pola aksara Lampung karena memiliki waktu komputasi yang lebih tinggi dibandingkan dengan metode SSD.

Kata kunci: aksara Lampung, *Sum of Absolute Differences* (SAD), *Sum of Squared Differences* (SSD), pengenalan pola, akurasi

ABSTRACT

COMPARISON OF THE ACCURACY OF SUM OF ABSOLUTE DIFFERENCES (SAD) AND SUM OF SQUARED DIFFERENCES (SSD) METHODS IN IDENTIFYING LAMPUNG SCRIPT PATTERNS

By

ANISA RAHMADINI

The Lampung script is a cultural heritage that holds historical and identity value for the Lampung community. However, in the digital era, its usage has significantly declined, necessitating preservation efforts through pattern recognition technology. This study compares the Sum of Absolute Differences (SAD) and Sum of Squared Differences (SSD) methods in identifying Lampung script patterns based on accuracy levels and computational efficiency. The dataset consists of 6,500 Lampung script characters, tested in three different dataset sizes (1,300, 3,900, and 6,500 characters) and two image resolutions (40×40 pixels and 80×80 pixels). The results indicate that both methods achieve nearly identical accuracy levels, with the highest accuracy of 90.54% obtained on the largest dataset using 80×80 pixel images. However, the SAD method consistently performs faster than the SSD method across all test scenarios. In the 6,500-character dataset, SAD requires an average processing time of 0.005 seconds, whereas SSD takes 0.534 seconds. Therefore, the SAD method is more recommended for implementing a Lampung script pattern recognition system due to its higher computational speed compared to the SSD method.

Keywords: Lampung script, Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), pattern recognition, accuracy

PERBANDINGAN TINGKAT AKURASI METODE *SUM OF ABSOLUTE DIFFERENCES* (SAD) DAN *SUM OF SQUARED DIFFERENCES* (SSD) DALAM IDENTIFIKASI POLA AKSARA LAMPUNG

Oleh
ANISA RAHMADINI

(Skripsi)

Sebagai Salah Satu Syarat Untuk Mencapai Gelar
SARJANA TEKNIK

Pada

Jurusan Teknik Elektro
Fakultas Teknik Universitas Lampung

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
2025

Judul Skripsi	:	**PERBANDINGAN TINGKAT AKURASI METODE *SUM OF ABSOLUTE DIFFERENCES* (SAD) DAN *SUM OF SQUARED DIFFERENCES* (SSD) DALAM IDENTIFIKASI POLA AKSARA LAMPUNG**
Nama Mahasiswa	:	**Anisa Rahmadini**
Nomor Pokok Mahasiswa	:	2015031019
Program Studi	:	Teknik Elektro
Fakultas	:	Teknik

MENYETUJUI

1. Komisi Pembimbing

Dr. Eng FX Arinto S.,S.T.,M.T	**Dr. Eng. Ageng Sadnowo R., S.T., M.T.**
NIP. 19691219 199903 1 002	NIP. 19690228 199803 1 001

2. Mengetahui

Ketua Jurusan Teknik Elektro	Ketua Program Studi S1 Teknik Elektro
Herlinawati, S.T., M.T.	**Sumadi, S.T., M.T.**
NIP. 19710314 199903 2 001	NIP. 19731104 200003 1 001

MENGESAHKAN

1. Tim Penguji

Ketua : **Dr. Eng FX Arinto S.,S.T.,M.T** .....

Sekretaris : **Dr. Eng. Ageng Sadnowo R., S.T., M.T.** .....

Penguji Utama : **Emir Nasrullah, S.T., M.Eng.** .....

2. Dekan Fakultas Teknik

Dr. Ing. Ir. Helmy Fitriawan, S.T., M.Sc.
NIP. 19750928 200112 1 002

Tanggal Lulus Ujian Skripsi : **15 Februari 2025**

SURAT PERNYATAAN

Saya menyatakan bahwa skripsi yang saya tulis ini merupakan hasil pemikiran saya sendiri, bukan merupakan hasil karya orang lain sebagai syarat memperoleh gelar sarjana. Adapun kutipan-kutipan yang terdapat dalam skripsi ini telah dicantumkan sumber-sumbernya dalam daftar pustaka sesuai dengan norma, kaidah, dan etika penulisan.

Apabila pernyataan ini terbukti tidak benar maka saya bersedia dikenakan sanksi sesuai dengan hukum yang berlaku.

Bandar Lampung, 16 April 2025

Rahmadini
2015031019

RIWAYAT HIDUP

ANISA RAHMADINI, dilahirkan di Bandung pada tanggal 1 Desember 2001. Anak kedua dari tiga bersaudara pasangan dari bapak Aam Samsudin dan ibu Ida Laila. Penulis melangsungkan pendidikan Sekolah Dasar (SD) yang diselesaikan di SD Negeri Pakansari 1 pada tahun 2014, Sekolah Menengah Pertama (SMP) di MTsN 1 Tanggamus pada tahun 2017, dan Sekolah Menengah Atas (SMA) di SMA Negeri 1 Kotaagung pada tahun 2020. Pada tahun 2020 penulis melanjutkan pendidikan di perguruan tinggi negeri, tepatnya di Universitas Lampung Fakultas Teknik pada Jurusan Teknik Elektro melalui jalur SNPMTN 2020 dan mendapatkan program beasiswa KIP-Kuliah.

Selama menjadi mahasiswa penulis aktif menjadi asisten praktikum Dasar Elektronika, Sistem Elektronika, dan Sistem Tertanam periode 2022 s.d. 2024. Penulis aktif di organisasi Himpunan Mahasiswa Teknik Elektro (HIMATRO) selama 2 kepengurusan sebagai bendahara Departemen Sosial dan Kewirausahaan (SOSWIR) dan menjadi anggota Kaderisasi dan Pengembangan Organisasi (KPO), serta aktif sebagai anggota Dewan Perwakilan Mahasiswa (DPM) Universitas Lampung. Penulis berhasil membawa Himatro Unila mendapatkan pendanaan pada kegiatan Program Penguatan Kapasitas Kemahasiswaan (PPK Ormawa) Kemendikbud Ristek pada tahun 2022. Pada bulan Juli 2023 penulis melaksanakan Kerja Praktik di PT PLN (Persero) Unit Induk Distribusi Lampung sub bidang Perencanaan Sistem Distribusi.

PERSEMBAHAN

Dengan Ridho Allah SWT

Dengan penuh penghormatan, aku panjatkan shalawat kepada Nabi Muhammad SAW. Karya tulis ini dengan sepenuh hati kupersembahkan kepada:

Ayah dan Ibuku Tercinta

Aam Samsudin dan Ida Laila

Serta Saudaraku Tersayang

Sukmawati Dewi, S.T. dan Ahmad Al-Rasyid

Dengan penuh rasa syukur, aku berterima kasih atas semua dukungan dan doa yang telah mengiringiku selama ini. Berkat itu semua, aku akhirnya dapat menyelesaikan karyaku dengan sebaik-baiknya.

MOTTO

"Live each day as if it were your last, and learn as if you were to live forever."
(Robin Sharma - *The Monk Who Sold His Ferrari*)

"Words are, in my not-so-humble opinion, our most inexhaustible source of magic. Capable of both inflicting injury and remedying it. Choose them carefully, for they shape the world in ways you cannot imagine."
(Albus Dumbledore - *Harry Potter and the Deathly Hallows*)

"Maybe life isn't about grand achievements, but about small joys, like eating your favorite food on a tough day."

(Baek Sehee - *I Want to Die But I Want to Eat Tteokbokki*)

"Tidak ada yang abadi dalam hidup ini, bahkan kesulitan pun akan berlalu. Jadi, tetaplah kuat, meskipun berat."

(Zee zee Aurora - *Tetaplah Kuat Meskipun Berat*)

SANWACANA

Segala puji dan syukur penulis panjatkan kepada Allah SWT atas segala limpahan rahmat, hidayah, dan inayah-Nya, sehingga penulisan skripsi yang berjudul **"Perbandingan Tingkat Akurasi Metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dalam Identifikasi Pola Aksara Lampung"** dapat diselesaikan dengan semestinya. Skripsi ini disusun sebagai bagian dari persyaratan untuk meraih gelar Sarjana Teknik pada Jurusan Teknik Elektro, Fakultas Teknik, Universitas Lampung.

Tak lupa, shalawat dan salam selalu penulis sampaikan kepada Nabi Muhammad SAW, keluarga, sahabat, serta para pengikutnya yang senantiasa berpegang teguh pada ajaran Islam hingga akhir zaman. Penyusunan skripsi ini tidak terlepas dari dukungan dan bantuan, baik berupa pemikiran maupun semangat, dari berbagai pihak. Dalam kesempatan ini penulis mengucapkan terimakasih kepada:

1. Ibu Prof. Dr. Ir. Lusmeilia Afriani, D.E.A., I.P.M., selaku Rektor Universitas Lampung.
2. Bapak Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc., selaku Dekan Fakultas Teknik Universitas Lampung.
3. Ibu Herlinawati, S.T., M.T. selaku Ketua Jurusan Teknik Elektro Fakultas Teknik Universitas Lampung.
4. Bapak Sumadi, S.T., M.T. selaku Ketua Program Studi Teknik Elektro Fakultas Teknik Universitas Lampung dan telah memberikan motivasi kepada penulis dalam menyelesaikan studi.
5. Bapak Dr. Eng FX Arinto Setyawan, S.T., M.T., selaku dosen pembimbing utama, penulis mengucapkan rasa terima kasih yang mendalam atas segala bantuan, bimbingan, dan dukungan yang telah diberikan selama proses penyusunan skripsi ini. Kesabaran dan ketulusan Bapak dalam membimbing, memberi arahan, serta menyemangati di setiap langkah telah menjadi dorongan

besar bagi penulis untuk terus berusaha menyelesaikan tugas ini dengan sebaik-baiknya. Nasihat dan ilmu yang Bapak berikan tidak hanya membantu penulis memahami materi, tetapi juga menjadi pelajaran berharga yang akan selalu dikenang sepanjang hidup. Terima kasih telah menjadi dosen yang begitu sabar dan penuh perhatian.

6. Bapak Dr. Eng. Ageng Sadnowo Repelianto, S.T., M.T., selaku dosen pembimbing pendamping dan dosen pembimbing akademik, penulis mengucapkan terima kasih yang sebesar-besarnya atas segala arahan, perhatian, dan kesabaran luar biasa selama proses bimbingan. Ketelitian Bapak dalam membantu memahami materi dan menyempurnakan penulisan, serta kesabaran Bapak dalam menghadapi setiap kesulitan penulis tanpa pernah menunjukkan rasa marah, telah menjadi dukungan yang sangat berarti. Bimbingan Bapak tidak hanya membantu penulis menyelesaikan skripsi ini dengan baik, tetapi juga menjadi inspirasi atas ketulusan dalam membagi ilmu.
7. Seluruh Dosen dan karyawan Jurusan Teknik Elektro Universitas Lampung, berkat ilmu yang telah diajarkan kepada penulis selama penulis menjalani masa studi di perkuliahan.
8. qDian Miranti, S.I.P, Zelika Marseria, S.Tr.Kes, Aulia Permata Sari, Amd.Farm, Atika Lestari, S.T, yang telah menemani penulis sejak usia 14 tahun dan menjadi teman yang tidak dapat digantikan oleh siapapun. Terimakasih atas telinga yang mendengar, bahu yang siap menjadi sandaran, dan tangan yang selalu sigap untuk memberi pertolongan.
9. Rachma Lingga Maulidya, S.T, Aymanul Fadillah, S.T, Adhiva Nur Fadheela, S.T, Anita Angraieni, S.T, Anna Zakkia Latifah, S.T. Terimakasih telah menjadi teman baik dan menjadi bagian dalam kehidupan penulis.
10. Muhammad Bimo Komala, S.T, terima kasih atas dukungan dan kesabaran yang selalu diberikan selama ini. Penulis menghargai segala perhatian dan usaha yang telah dilakukan, yang membuat perjalanan ini menjadi lebih mudah.
11. Kakak-kakak ku, Kak Mutia, Kak Ahlul, dan Muhammad Farhan Deros yang telah banyak membantu memberi arahan dan motivasi dalam mengerjakan skripsi ini.

12. Keluarga besar di Laboratorium Teknik elektronika, Adzom, Ryandi, Ridwan, Rahmat, Andres, Mahesa (Dudut), Azra, Steveen, Wahyudi, Faisal, Wildhan, Haqqu, Hud, Farda, dan Esa yang selalu memberikan dukungan, pertolongan, canda tawa, dalam setiap proses apapun selama menjadi asisten laboratorium teknik elektronika.
13. Keluarga Deros (Kader 20), penulis mengucapkan terima kasih yang sebesar-besarnya atas kebersamaan yang begitu berarti selama masa perkuliahan. Terima kasih telah menjadi teman seperjuangan yang selalu ada di setiap langkah, berbagi tawa dan canda di saat suka, serta menjadi sandaran di saat duka. Momen-momen kebersamaan, dukungan, dan kekuatan yang kalian berikan telah menjadi bagian penting dari perjalanan ini.
14. Keluarga besar HIMATRO, HELLIOS 20 yang telah memberikan banyak motivasi, nilai-nilai sosial, dan bantuan dalam berbagai hal.
15. Terkhusus Bagas Chandra, Mahesa Yudhistira, Muhammad Haqqu, Kadapi, dan Rahmat Pribadi Putra yang telah berkenan meminjamkan laptopnya pada penulis sehingga penulis dapat menyelesaikan skripsi ini.
16. Semua pihak yang tidak dapat disebutkan satu persatu dan terlibat langsung maupun tidak langsung yang telah membantu penulis dalam pembuatan skripsi.

Semoga Allah SWT membalas semua perbuatan dan kebaikan yang telah diberikan kepada Penulis sampai dengan terselesaikannya Skripsi ini. Penulis menyadari bahwa laporan skripsi ini masih memiliki banyak kekurangan, baik dari segi penyusunan maupun pemilihan kata. Maka dari itu penulis terbuka untuk menerima masukan kritik dan saran yang dapat membangun penulis kedepannya. Semoga penulisan skripsi ini dapat bermanfaat bagi pembaca.

Bandar Lampung, 16 April 2025
Penulis,

Anisa Rahmadini

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pelestarian aksara Lampung menjadi semakin penting seiring dengan perkembangan era digital dan globalisasi. Bahasa dan aksara adalah fondasi utama yang membawa nilai-nilai budaya, sejarah, dan identitas suatu masyarakat. Masyarakat Lampung, dengan aksara Lampung-nya yang unik, memiliki warisan budaya yang merefleksikan kearifan lokal, cerita leluhur, serta kekayaan budaya yang harus dilindungi dan dipertahankan. Aksara kaganga lampung merupakan salah satu aset budaya indonesia dari suku asli lampung. Saat ini hanya sedikit masyarakat yang memahami aksara kaganga lampung ini[1]. Dalam konteks ini, pengajaran aksara Lampung bukan hanya sebagai upaya pelestarian, melainkan juga sebagai langkah strategis untuk menjaga keberlanjutan bahasa dan budaya lokal di tengah arus globalisasi yang semakin kuat. Pendidikan aksara Lampung menjadi krusial dalam mendukung pemahaman, penggunaan, dan pelestarian aksara tersebut. Peran teknologi dalam upaya ini sangat signifikan, karena dapat membantu mengatasi tantangan dan membangun solusi yang efektif untuk mempertahankan budaya tersebut.

Dalam usaha pelestarian aksara Lampung, metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD), memberikan langkah inovatif untuk mengukur tingkat kemiripan antara karakter aksara Lampung. Dengan kemajuan teknologi ini, pengajaran aksara Lampung dapat lebih mudah diintegrasikan ke dalam berbagai platform digital, sehingga mempermudah proses pelestarian budaya melalui media modern. Aksara Lampung memiliki keunikan dalam hal perbedaan skala, variasi bentuk karakter, dan kompleksitas bahasa yang memerlukan solusi

teknis yang tepat. Dalam upaya mengenali dan mengajarkan aksara Lampung melalui teknologi, metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) menawarkan pendekatan efektif dalam proses pengenalan karakter aksara. Kedua metode ini memiliki dasar yang serupa, yaitu mengukur tingkat kemiripan antara karakter pada citra digital dengan menghitung perbedaan intensitas piksel antar karakter. Kesamaan utama dari metode SAD dan SSD terletak pada konsep mereka yang sama-sama mengakumulasikan perbedaan piksel, namun dengan pendekatan yang sedikit berbeda yaitu SAD menghitung jumlah perbedaan absolut, sementara SSD menghitung jumlah kuadrat perbedaan.

Pentingnya metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dalam identifikasi pola aksara Lampung tidak hanya terletak pada tingkat akurasi, tetapi juga pada efisiensi waktu komputasi. Dengan menerapkan solusi teknologi yang tepat, pengajaran dan konversi aksara Lampung dapat dilakukan secara lebih efisien, meminimalkan kesalahan pengenalan pola aksara, dan memastikan hasil yang optimal. Ini menjadi langkah maju dalam mendukung pelestarian aksara Lampung, karena dapat diakses dengan mudah melalui berbagai media digital, yang pada akhirnya memperluas jangkauan pendidikan budaya lokal ini.

Dengan demikian, penerapan metode SAD dan SSD dalam identifikasi pola aksara Lampung tidak hanya memberikan solusi teknis untuk pengajaran aksara, tetapi juga menjadi bagian penting dari pelestarian bahasa dan budaya Lampung di era digital. Metode ini diharapkan dapat memberikan dampak positif dalam mempertahankan keberlanjutan budaya lokal serta menghubungkan tradisi dengan inovasi teknologi. Pendidikan digital aksara Lampung dengan penerapan metode ini adalah solusi yang tidak hanya praktis, tetapi juga penting untuk menjaga keberlanjutan nilai-nilai budaya dalam konteks global yang semakin modern

1.2 Tujuan Penelitian

Adapun tujuan yang ingin dicapai pada penelitian ini adalah sebagai berikut.

1. Mengukur tingkat akurasi dari metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dalam identifikasi pola Aksara Lampung.

2. Menganalisis dan membandingkan tingkat akurasi metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dalam pengenalan pola aksara Lampung pada berbagai ukuran citra.

1.3 Rumusan Masalah

Adapun rumusan masalah pada penelitian ini adalah sebagai berikut

1. Apa pengaruh perbedaan pendekatan matematis antara SAD dan SSD terhadap tingkat kesalahan dalam identifikasi pola aksara Lampung?
2. Bagaimana perbandingan tingkat akurasi dan efisiensi antara metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dalam pengenalan pola aksara Lampung pada berbagai ukuran citra?

1.4 Batasan Masalah

Adapun batasan masalah yang ditetapkan dalam penelitian ini adalah sebagai berikut.

1. Penelitian ini dibatasi pada identifikasi pola aksara Lampung tanpa mempertimbangkan aksara lain.
2. Penelitian ini tidak melibatkan analisis bahasa natural atau pemahaman konteks budaya, hanya berfokus pada kemiripan visual antara karakter aksara Lampung.
3. Batasan metode hanya pada penggunaan metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD), tanpa mempertimbangkan pendekatan atau algoritma lainnya.

1.5 Manfaat Penelitian

Adapun manfaat yang didapatkan pada penelitian ini adalah sebagai berikut.

1. Penelitian ini diharapkan memberikan kontribusi signifikan dalam meningkatkan akurasi dan efisiensi metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD), hasil penelitian ini dapat menjadi dasar yang kuat untuk mendukung pelestarian aksara Lampung.

2. Hasil penelitian yang menganalisis tingkat akurasi dan efisiensi waktu metode SAD dan SSD dalam pengenalan pola aksara Lampung diharapkan dapat menjadi landasan teknis bagi pengembangan aplikasi.
3. Penelitian ini berkontribusi dalam mengevaluasi keunggulan dan keterbatasan metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dalam identifikasi pola aksara Lampung.

1.6 Hipotesis

Penggunaan metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dapat mengidentifikasi pola aksara Lampung dengan akurasi yang optimal serta efisiensi waktu komputasi yang memadai, sehingga mendukung pelestarian aksara Lampung dalam konteks digital. Selain itu, kedua metode ini menghasilkan perbedaan tingkat akurasi dalam identifikasi pola aksara Lampung, dimana masing-masing metode memiliki kelebihan dan kekurangan dalam menangani perbedaan dan kesamaan antar pola. Dengan demikian, perbandingan antara SAD dan SSD akan memberikan wawasan mengenai metode yang lebih efektif untuk pengenalan aksara Lampung dalam aplikasi digital.

1.7 Sistematika Penulisan

Sistematika penulisan yang digunakan pada penelitian tugas akhir ini adalah sebagai berikut.

BAB I PENDAHULUAN

Penelitian ini dilatarbelakangi oleh pentingnya pengajaran aksara Lampung sebagai langkah strategis untuk menjaga keberlanjutan bahasa dan budaya lokal di tengah perubahan zaman. Keunikan aksara Lampung, seperti perbedaan skala dan variasi bentuk karakter, diatasi dengan respons tajam dari metode SAD dan SSD.

BAB II TINJAUAN PUSTAKA

Pada bab ini menjelaskan tentang aksara Lampung, citra analog dan digital, serta konsep pengolahan citra digital. Pembahasan mencakup metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD). Terakhir, bab ini

mengenalkan platform *Visual Studio Code,* bahasa pemrograman *Python*, dan beberapa *Library* seperti *OpenCV* dan *Numpy* yang relevan untuk pengolahan citra.

BAB III METODOLOGI PENELITIAN

Pada bab ini akan menjelaskan terkait alat dan bahan yang digunakan pada penelitian terkait. Pada penelitian ini, metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) digunakan untuk mengidentifikasi karakter aksara Lampung dalam citra. Keseluruhan sistem akan dijelaskan melalui diagram alir dan diagram blok *system*.

BAB IV HASIL DAN PEMBAHASAN

Bab ini akan menjelaskan tentang data aksara Lampung yang diperoleh melalui angket penulisan aksara Lampung, yang akan ditulis oleh beberapa individu secara acak. Informasi yang terdapat dalam data ini akan diolah menggunakan teknik, metode, dan ketentuan yang telah ditetapkan dalam kerangka penelitian. Hasil dari pengolahan data tersebut kemudian dianalisis dengan merujuk pada tujuan penelitian, rumusan masalah, dan batasan masalah yang telah ditentukan sebelumnya. Proses ini dilakukan dengan tujuan untuk memahami dan menginterpretasikan pola penulisan aksara Lampung sesuai dengan landasan metodologi yang digunakan dalam penelitian.

BAB V KESIMPULAN DAN SARAN

Bab ini akan menjelaskan terkait temuan dari data yang telah diolah dan dianalisis sesuai dengan hasil yang diperoleh pada bab sebelumnya. Temuan dari penelitian akan dijelaskan secara singkat, diikuti dengan rangkuman hasil penelitian dan pembahasan mengenai saran untuk penelitian selanjutnya berdasarkan kekurangan yang teridentifikasi dari hasil penelitian.

DAFTAR PUSTAKA

LAMPIRAN

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Eliza Hara dkk (2016) dalam penelitiannya yang berjudul "Penggunaan Deteksi Tepi (*Canny*) pada Sistem Pengenalan Tulisan Tangan Aksara Lampung Berbasis Jaringan Syaraf Tiruan" menerapkan model *backpropogation* dan deteksi tepi *canny* untuk membuat aplikasi pengenalan aksara Lampung. Penelitian ini bertujuan untuk membuat aplikasi pengenalan tulisan tangan aksara Lampung. Sistem pengenalan tulisan tangan aksara Lampung dikembangkan dengan teknik pengolahan citra dan jaringan syaraf tiruan *backpropogation*[2].

Aryantio dkk (2015) dalam penelitian yang berjudul "Pengenalan Aksara Lampung Menggunakan Jaringan Syaraf Tiruan" membuat aplikasi yang akan digunakan untuk mengenali pola aksara Lampung. Pengenalan pola terdiri dari pemindaian citra, pengolahan awal citra, ekstraksi fitur dan klasifikasi. Proses klasifikasi dilakukan dengan menggunakan jaringan syaraf tiruan propagasi balik. Proses klasifikasi dilakukan oleh jaringan syaraf tiruan. Jaringan syaraf tiruan yang dibentuk memiliki 2 jaringan syaraf tiruan yaitu jaringan syaraf tiruan utama dan jaringan syaraf tiruan karakter[3].

A. W. Mahastama dkk (2016) dalam penelitian nya yang berjudul "Perbandingan Algoritma *Sum of Squared Difference* (SSD) dan *Optimised Sum of Absolute Difference* (OSAD) untuk Pengenalan Simbol Pada Citra Ekspresi Matematika Tercetak" melakukan penelitian untuk mengkaji terutama pada pengenalan simboll pada ekspresi matematika dengan pendekatan *template matching* berbasis korelasi, menggunakan dua metode yang diperbandingkan yaitu

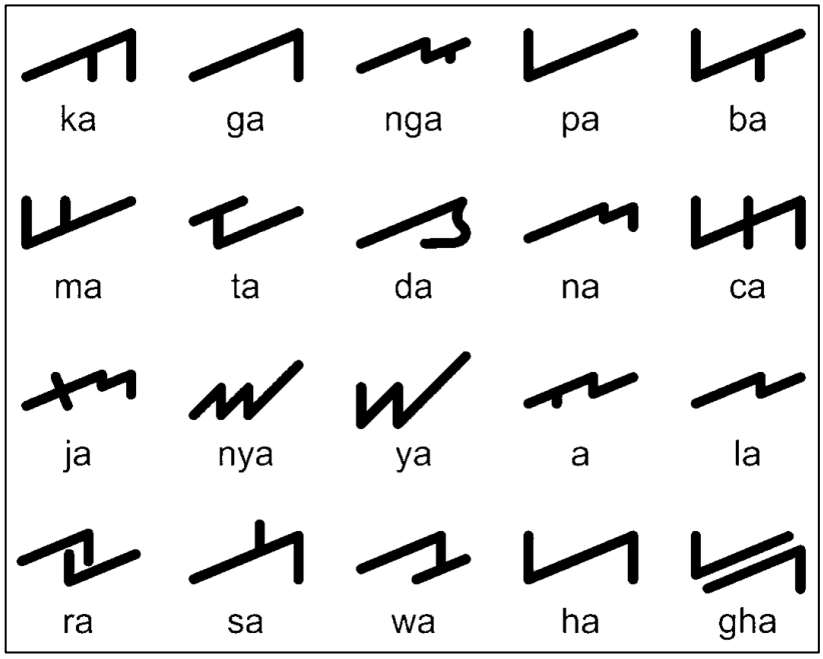
Sum of Squared Difference (SSD) sebagai salah satu metode yang telah banyak digunakan dalam *template matching*, serta algoritma *Optimised Sum of Absolute Difference* sebagai salah satu algoritma yang dikembangkan baru-baru ini dan menggunakan prinsip *average face* yaitu sebuah pola rata-rata yang mewakili seluruh pola yang merepresentasikan sebuah simbol, sehingga dapat menghemat waktu pencocokan[4].

2.2 Aksara Lampung

Aksara Lampung, atau yang dikenal sebagai Had Lampung, merupakan sistem tulisan yang digunakan di wilayah Lampung[5]. Aksara Lampung melibatkan elemen-elemen seperti huruf dasar, turunan huruf, kombinasi huruf ganda, serta kelompok konsonan. Aksara Lampung, yang sering disebut sebagai "Kaganga" ditulis dan dibaca dari kiri ke kanan, dengan jumlah total huruf induk sebanyak 20 buah.

2.2.1 Induk Huruf atau Kelabai Sukhat

Induk huruf atau kelabai sukhat pada aksara Lampung berjumlah 20 huruf yang diantara nya dapat dilihat pada Gambar 2.1

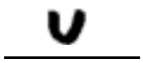
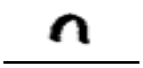
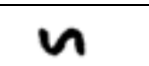
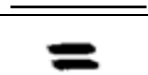
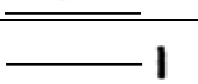
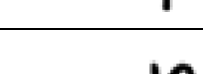
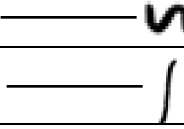
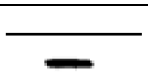
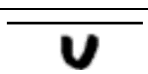


Gambar 2. 1 Induk Huruf atau Kelabai Sukhat dalam Aksara Lampung

2.2.2 Anak Huruf atau Anak Sukhat

Dalam aksara Lampung, terdapat huruf induk dan 12 anak huruf yang ditempatkan di sekitar induk huruf. Adapun penempatan daripada anak huruf aksara Lampung digambarkan pada Tabel 2.1

Tabel 2.1 Anak Huruf atau Anak Sukhat dalam Aksara Lampung

NO	Nama	Bunyi	Tanda	Contoh
1.	Ulan	i		hati
2.	Ulan	e		menara
3.	Bicek	e		cabe
4.	Rejunjung	r		cacar
5.	Tekelubang	ng		batang
6.	Datas	n		makan
7.	Tekelingai	ai		bangkai
8.	Keleniah	ah		sudah
9.	Nengen	-		tutup/
10.	Bitan	o		bodoh
11.	Bitan	u		buncit
12.	Tekelungau	au		limau

Adapun rincian daripada anak huruf pada aksara Lampung adalah sebagai berikut

1. Ulan adalah tanda kecil berbentuk setengah lingkaran yang ditempatkan di atas huruf utama. Ulan terdiri dari dua jenis: ulan yang menghadap ke atas melambangkan bunyi **i** sedangkan yang menghadap ke bawah berbunyi **e**
2. Bicek adalah tanda kecil di atas huruf utama berbentuk garis lurus vertikal yang berbunyi **e**

3. Rejunjung adalah tanda berbentuk spiral di atas huruf utama yang menghasilkan bunyi **r**
4. Tekelubang adalah tanda berbentuk garis lurus di atas huruf utama yang melambangkan bunyi **ng**
5. Datas adalah tanda berupa dua garis mendatar (seperti tanda sama dengan) di atas huruf utama yang menghasilkan bunyi **n**
6. Tekelingai adalah tanda berbentuk mirip huruf "h" yang terletak di depan atau di sebelah kanan huruf utama dan menghasilkan bunyi **ai**
7. Keleniah adalah tanda berbentuk garis lurus vertikal yang ditempatkan di depan huruf utama dan menghasilkan bunyi **ah**
8. Nengen adalah tanda berupa garis miring di depan huruf utama yang berfungsi sebagai penanda penghentian bunyi huruf. Namun, nengen tidak berlaku untuk beberapa huruf seperti "n," "y," "h," dan "w," yang bisa menggunakan tanda khusus lainnya.
9. Bitan adalah tanda yang berada di bawah huruf utama, terdiri dari dua jenis: garis pendek mendatar yang berbunyi **au** dan garis tegak yang berbunyi **o**
10. Tekelungau adalah tanda di bawah huruf utama, berbentuk setengah lingkaran kecil yang menghadap huruf utama dan melambangkan bunyi **au**

2.3 Citra

Citra, menurut definisi dalam kamus Webster, mengacu pada representasi, kemiripan, atau imitasi dari suatu objek. Sebagai contoh, sebuah foto apel dapat mencerminkan identitas buah apel tersebut ketika difoto dengan kamera. Citra dapat berbentuk hasil fotografi, lukisan, atau representasi lainnya yang terjadi di berbagai media seperti kertas, kanvas, dan layar monitor. Dalam konteks yang lebih teknis, citra juga dapat diartikan sebagai distribusi variasi antara gelap-terang, redup-cerah, dan/atau beragam warna di suatu bidang data. Secara formal, konsep citra dapat diungkapkan melalui angka-angka yang merepresentasikan variasi intensitas kecerahan dan/atau warna dalam arah horizontal dan vertikal. Secara umum, citra dapat dikelompokkan menjadi dua jenis, yaitu citra yang bersifat kontinyu dan citra yang bersifat diskrit[6].

2.3.1 Citra Analog

Citra analog merujuk pada gambar yang memanfaatkan data analog sebagai *input*nya. Dalam hal ini, data analog tidak direpresentasikan secara digital di dalam komputer, melainkan mencerminkan fakta sebagaimana adanya seperti tampilan pada layar televisi, hasil foto sinar-X, gambar yang dicetak pada kertas foto, lukisan, pemandangan, *output* dari *CT scan*, gambar yang direkam pada pita kaset, dan sebagainya. Citra analog tidak dapat direpresentasikan secara langsung dalam bentuk digital di komputer.

2.3.2 Citra Digital

Citra digital adalah rangkaian piksel yang tersusun dalam larik dua dimensi, dengan titik asal (0,0) terletak di sebelah kiri atas citra[7]. Penentuan lokasi titik asal ini mengacu pada cara penulisan matriks dalam pemrograman komputer, berbeda dengan koordinat grafik. Karena citra digital memiliki bentuk matriks, manipulasinya juga mengikuti prinsip matriks. Contoh dari citra digital dapat dilihat pada Gambar 2.4

Gambar 2.2 Representasi Citra Digital ‘Lena’

Citra digital dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra[8]. Pengolahan citra digital adalah cabang ilmu komputer yang berkutat pada usaha untuk melakukan

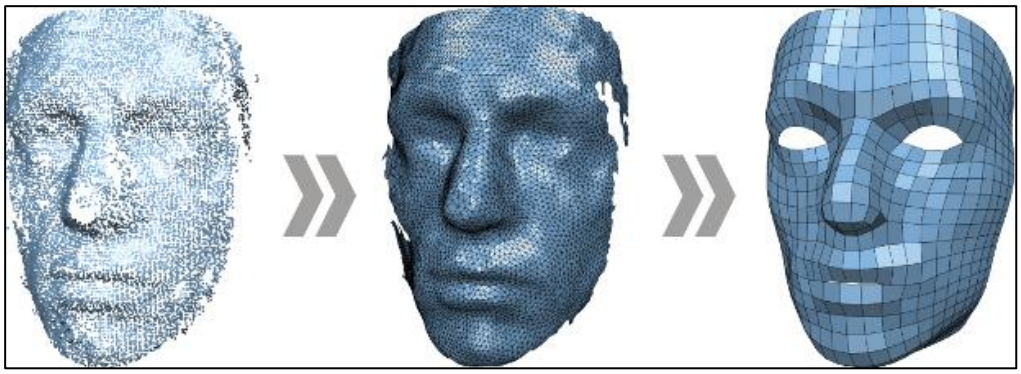
transformasi[9]. Pengolahan citra digital melibatkan berbagai operasi, seperti peningkatan kualitas citra, pemulihan citra, dan perangkat lunak pengindraan jauh.

2.4 Pengolahan Citra

Pengolahan, menurut definisi dari Kamus Besar Bahasa Indonesia (KBBI), merujuk pada suatu metode atau proses untuk mengubah sesuatu agar menjadi berbeda atau lebih sempurna. Sementara itu, citra dalam konteks KBBI diartikan sebagai rupa atau gambar, khususnya gambar yang diperoleh melalui sistem visual. Dengan menggabungkan kedua konsep ini, pengolahan citra dapat diartikan sebagai suatu upaya untuk merubah citra menjadi bentuk yang lebih sempurna atau sesuai dengan yang diinginkan. Secara sederhana, pengolahan citra adalah proses yang melibatkan citra sebagai *input* dan menghasilkan citra *output* sesuai dengan keinginan. Pada awalnya, pengolahan citra digital digunakan terutama untuk mengonversi citra analog ke format digital dan untuk meningkatkan kualitas citra.

Namun, seiring dengan perkembangan teknologi, penggunaan pengolahan citra digital menjadi lebih beragam. Algoritma pengolahan citra digunakan untuk menggantikan fungsi sensor penglihatan manusia dengan sensor buatan, seperti kamera. Kemajuan dalam waktu komputasi memungkinkan pengolahan citra digital dilakukan secara *real-Time*. Selain itu, perkembangan memori memfasilitasi pengkodean citra analog menjadi citra warna digital yang mendekati warna aslinya. Dengan demikian, pengolahan citra digital mencakup sejumlah aplikasi dan perkembangan teknologi yang memungkinkan manipulasi citra secara lebih efektif dan sesuai dengan kebutuhan. Pengolahan citra digital memiliki berbagai aplikasi, termasuk dalam bidang *computer vision* dan pengenalan pola. Ini melibatkan konsep-konsep seperti representasi citra, transformasi citra, dan analisis citra[10][8].

Teknik-teknik dalam pengolahan citra digital mencakup pengolahan citra analog dan digital, serta metode seperti penguatan citra, restorasi citra, segmentasi citra, dan kompresi citra. Citra digital terdiri dari larik (*array*) yang berisi nilai-nilai intensitas cahaya[7]. Representasi citra digital terdiri dari piksel-piksel yang memiliki koordinat dan nilai intensitas warna diperlihatkan pada Gambar 2.5.



Gambar 2. 3 Bentuk Pengolahan Citra Digital

2.5 *Sum of Absolute Differences* (SAD)

Algoritma SAD (*Sum of Absolute Differencess*) digunakan untuk mengukur kemiripan antara gambar dengan menghitung perbedaan absolut antara piksel-piksel gambar (gambar *template*) dan piksel – piksel yang sesuai (gambar pencarian) dalam blok makro. Perbedaan – perbedaan ini kemudian dijumlahkan untuk menghasilkan blok kemiripan[11]. Algoritma ini hanya memerlukan dua operasi matematika dasar, yaitu penjumlahan dan pergeseran. Algoritma SAD (*Sum of Absolute Differencess*) adalah metrik yang paling sederhana yang mempertimbangkan semua piksel dalam blok untuk perhitungan, dan juga secara terpisah, sehingga memudahkan implementasinya dan dapat dijalankan secara paralel. Karena kesederhanaannya, algoritma ini merupakan salah satu yang paling cepat dan dapat digunakan secara luas dalam estimasi gerakan blok dan pengenalan objek. SAD diperoleh dari perhitungan matriks selisih. Formula matematis-nya dapat diwakili oleh persamaan 2.1

$$SAD = \sum_i \sum_j |A(i,j) - B(i,j)| \quad (2.1)$$

Dimana,

d(A,B) = selisih jarak fitur matriks A dan B

A(i,j) = komponen fitur matriks A pada kolom i dan baris j

B(i,j) = komponen fitur matriks B pada kolom i dan baris j

Fungsi dari SAD (*Sum of Absolute Differences*) mengukur perbedaan antara nilai piksel. Disparitas dihitung di setiap piksel dalam gambar dan untuk setiap

kemungkinan disparitas. Fungsi ini menjumlahkan intensitas semua piksel di sekitar untuk setiap piksel dalam gambar kiri. Selisih absolut antara jumlah ini dan jumlah piksel beserta piksel sekitarnya dalam gambar kanan kemudian dihitung.

Nilai SAD (*Sum of Absolute Differencess*) yang lebih kecil menunjukkan tingkat kemiripan yang lebih tinggi antara kedua blok piksel. SAD (*Sum of Absolute Differencess*) sering digunakan dalam berbagai aplikasi pengolahan citra, terutama dalam konteks pemrosesan video dan pengenalan pola. Nilai SAD yang lebih kecil menunjukkan kemiripan antara gambar uji dan *template*. Untuk menyederhanakan proses dalam rangkaian digital, gerbang XOR dapat digunakan untuk menentukan apakah setiap piksel antara gambar *template* dan gambar sumber berbeda atau tidak. Perhitungan SAD dilakukan pada setiap piksel mulai dari kiri atas hingga kanan bawah gambar sumber. Hasilnya dibandingkan dengan nilai ambang.

2.6 *Sum of Squared Differences* (SSD)

Sum of Squared Differences (SSD) adalah salah satu metode pengukuran kemiripan yang berdasarkan perbedaan intensitas piksel ke piksel antara dua gambar[12]. Metode ini menghitung jumlah kuadrat dari hasil perkalian selisih piksel antara dua gambar. Dengan pengukuran kemiripan ini, titik pencocokan dapat ditentukan dengan mempertimbangkan lokasi nilai minimum dalam matriks gambar[13]. Secara umum, SSD dapat menggunakan formulasi MSE tanpa menggunakan normalisasi yang ditunjukkan pada Persamaan 2.2

$$SSD = \sum_i \sum_j \big(A(i,j) - B(i,j)\big)^2 \quad (2.2)$$

Dimana,

d(A,B) = selisih jarak fitur matriks A dan B

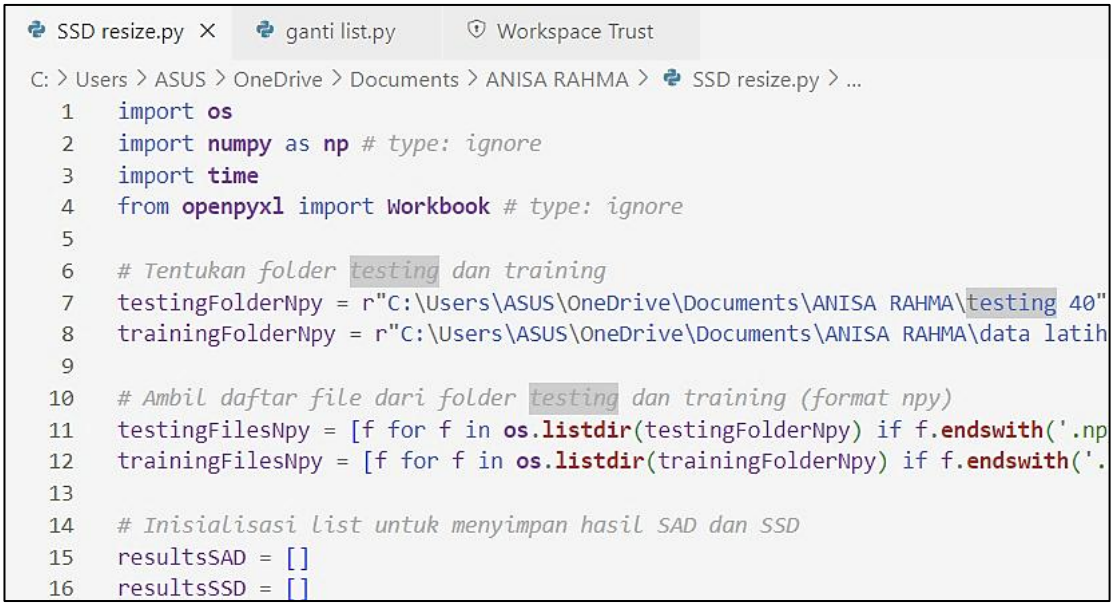
A(i,j) = komponen fitur matriks A pada kolom i dan baris j

B(i,j) = komponen fitur matriks B pada kolom i dan baris j

Penerapan SSD sering digunakan dalam pencocokan *template* di mana gambar *input* digulung di atas gambar *template*, dan SSD dihitung pada setiap posisi. Semakin kecil nilai SSD, semakin besar kemiripan antara gambar *input* dan *template*[14].

2.7 *Visual Studio Code*

Visual Studio Code (VS *Code*) adalah editor teks dan lingkungan pengembangan terpadu (IDE) yang dikembangkan oleh *MicrosOft* dan dirilis pada tahun 2015. VS *Code* dikenal karena fleksibilitas dan kinerjanya yang ringan, serta dukungan lintas platform (*Windows, macOS,* dan *Linux*). Meskipun sederhana, VS *Code* dilengkapi dengan fitur canggih seperti *IntelliSense* untuk *auto-completion*, integrasi Git untuk kontrol versi, serta debugging untuk membantu pengembangan dan pengujian aplikasi.



Gambar 2.4 Tampilan Visual Studio Code

2.8 *Python*

Python adalah bahasa pemrograman tingkat tinggi yang dikenal karena sintaksisnya yang sederhana, mudah dibaca, dan fleksibel. Diciptakan oleh Guido van Rossum pada awal tahun 1990-an, *Python* dirancang untuk memudahkan pengembangan perangkat lunak dan mempromosikan pemrograman yang produktif. Bahasa ini mendukung paradigma pemrograman berorientasi objek, fungsional, dan pemrograman imperatif. Kelebihan utama *Python* adalah kemampuannya yang sangat baik dalam pengolahan string, manipulasi data, dan pengembangan aplikasi web. *Python* juga menyediakan sejumlah besar perpustakaan standar dan pustaka pihak ketiga yang memperluas fungsionalitasnya,

seperti *Numpy* dan Pandas untuk pengolahan data, *TensorFlow* dan *PyTorch* untuk pembelajaran mesin, serta Django dan Flask untuk pengembangan web.

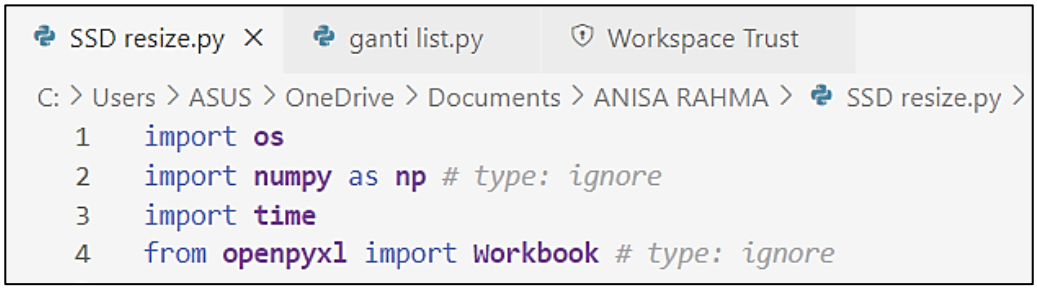
Gambar 2.5 Logo Software Python

2.8.1 *Library OpenCV (cv2)*

OpenCV, atau *Open Source Computer Vision Library*, adalah pustaka perangkat lunak open-source yang berfokus pada pengolahan citra dan pengenalan pola. Dirancang untuk mendukung berbagai aplikasi komputer *vision*, *OpenCV* menyediakan algoritma-algoritma kunci untuk membaca, memanipulasi, dan menganalisis gambar serta video. Fungsinya mencakup deteksi objek, pengenalan wajah, segmentasi gambar, dan banyak lagi. *OpenCV* sering menjadi pilihan utama dalam proyek-proyek yang melibatkan pengolahan citra dan visi komputer karena ketersediaan kode sumber terbuka dan kelengkapan fiturnya.

2.8.2 *Numpy*

Numpy, atau *Numerical Python*, merupakan pustaka dasar dalam ekosistem *Python* untuk komputasi numerik. *Numpy* menawarkan struktur data *array* yang efisien dan fungsi matematika yang kuat, memungkinkan pengguna untuk melakukan operasi matematika dan statistika dengan mudah pada data numerik. *Numpy* memungkinkan operasi vektorisasi yang efisien, yang sangat berguna dalam pengolahan data numerik yang melibatkan besar *dataset*.



Gambar 2.6 *Library OpenCV*

BAB III

METODOLOGI PENELITIAN

3.1 Waktu dan Tempat

Adapun penelitian ini dilakukan di Laboratorium Teknik Elektronika, Laboratorium Terpadu Jurusan Teknik Elektro, Universitas Lampung, dimulai pada bulan Maret 2024 sampai dengan bulan Februari 2025.

3.2 Alat dan Bahan

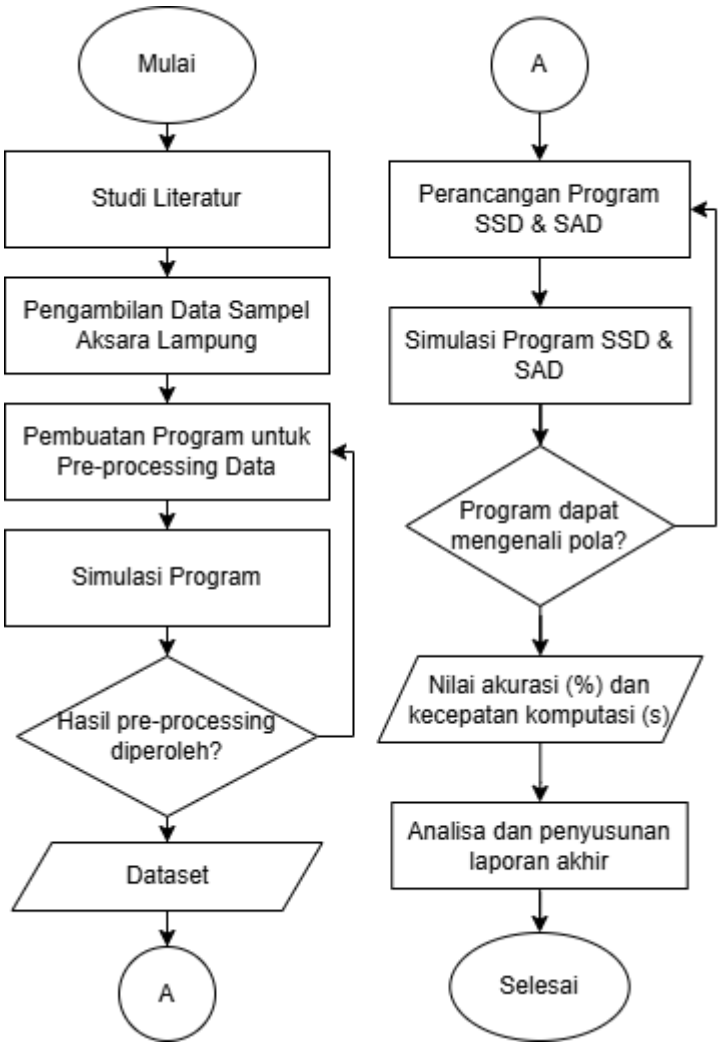
Untuk mendukung penelitian digunakan beberapa alat dan bahan sebagai berikut:

Tabel 3.1 Alat dan Bahan

Alat dan Bahan	Justifikasi Penggunaan
Laptop Acer Espire, *windows 10 pro, Intel(R) Core(TM)*	Sebagai tempat penyusunan program, dan simulasi.
Angket Aksara Lampung	Sebagai objek utama yang dijadikan sebagai data penelitian.
Software Python	Bahasa pemograman yang digunakan untuk penelitian.
HP Deskjet GT 5820 (printer, scan)	Sebagai alat untuk *scanning* data tulisan tangan aksara Lampung
Software Visual Studio Code	Aplikasi yang digunakan sebagai kode editor.

3.3 Diagram Alir Penelitian

Adapun tahapan yang dilakukan pada penelitian ini digambarkan melalui diagram alir pada Gambar 3.1.

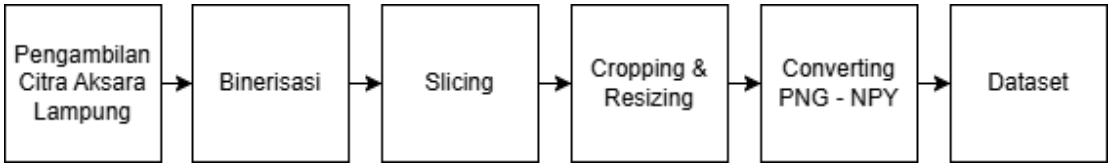


Gambar 3.1 Diagram Alir Penelitian

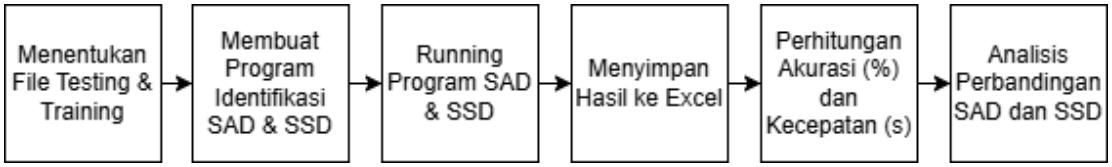
Penelitian ini dimulai dengan studi literatur untuk membangun dasar teoritis dengan merujuk pada penelitian-penelitian terdahulu. Pengumpulan data dilakukan dengan cara mengumpulkan data tulisan aksara Lampung yang dituliskan oleh beberapa orang dengan pola aksara yang beragam yang kemudian akan digunakan sebagai data sampel pada rancangan program dengan menggunakan *Visual Studio Code.* Selanjutnya akan dilakukan simulasi pada program *pre-processing* yang telah dibuat, apabila simulasi yang dijalankan berhasil, maka akan diperoleh data hasil yang akan dijadikan sebagai data *template* (*template*).

Dengan demikian penelitian dapat dilanjutkan ke tahap berikutnya, yaitu merancang atau menyusun program untuk pengenalan pola menggunakan metode *Sum of Absolute Differences* dan *Sum of Squared Differences*. Program dirancang sekaligus sehingga dalam satu program akan menghitung SAD dan SSD secara bersamaan. Setelah program selesai dibuat, dilakukan uji coba untuk menguji program tersebut. Hasil dari kedua metode, berupa angka yang menunjukkan perbedaan piksel serta waktu komputasi akan dianalisis untuk menilai tingkat akurasinya.

3.4 Diagram Blok Perancangan Sistem



(a) Diagram Blok *Pre-processing*



(b) Diagram Blok Sistem Identifikasi

Gambar 3.2 Diagram Blok Perancangan Sistem Identifikasi

Proses pengolahan citra dimulai dengan pengambilan Citra Aksara Lampung dalam bentuk gambar digital yang kemudian dibinerisasi menjadi citra biner. Selanjutnya, dilakukan *slicing* untuk memisahkan karakter aksara, diikuti oleh *cropping* dan *resizing* untuk menyesuaikan ukuran citra. Setelah itu, citra dikonversi dari format PNG ke NPY (*Numpy array*) agar dapat digunakan dalam pemrosesan lebih lanjut. Citra yang telah diolah digunakan sebagai data hasil, kemudian ditentukan folder *testing* dan *training* untuk memisahkan data uji dan *template*. *File* dari kedua folder diambil untuk menjalankan program yang menghitung nilai SAD (*Sum of Absolute Differences*) dan SSD (*Sum of Squared Differences*) antara citra uji dan *template*. Program kemudian dijalankan untuk menghitung nilai SAD dan SSD, dan hasilnya disimpan ke *file* Excel. Pada tahap akhir, tingkat akurasi dan waktu rata-rata dari algoritma akan dihitung. Hasil

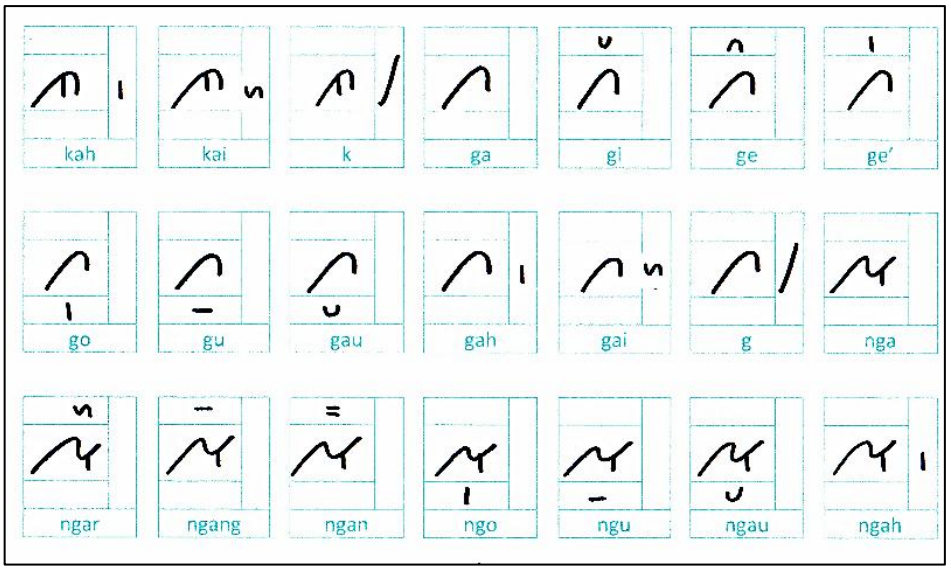
pengukuran ini akan dibandingkan antara *Sum of Absolute Difference* (SAD) dan *Sum of Squared Difference* (SSD) untuk menentukan metode yang paling optimal dalam hal akurasi dan efisiensi waktu. Perbandingan ini diharapkan dapat memberikan wawasan lebih mendalam tentang performa kedua metode dalam aplikasi pengenalan pola.

3.4.1 Pre-*processing*

Dalam tahap *pre-processing* ini, serangkaian langkah dilakukan sebelum memasuki tahap konversi. Tahap *preprocessing* dalam penelitian ini mencakup beberapa langkah, yaitu sebagai berikut.

a) Pengambilan Citra

Pengambilan data citra dilakukan dengan menggunakan scanner HP Deskjet GT 5820, dimana 30 angket yang telah diisi dengan aksara Lampung oleh responden diletakkan rata pada kaca *scanner*. Setelah itu, penutup *scanner* ditutup dengan hati-hati, dan proses pemindaian dilakukan melalui perangkat lunak dengan penyesuaian resolusi sesuai kebutuhan. Setelah pengaturan selesai, pemindaian dimulai dengan menekan tombol *Scan*. Adapun hasil pengambilan citra ditunjukkan pada gambar 3.3



Gambar 3.3 Hasil Scanning Angket Aksara Lampung

b) Binerisasi

Proses binarisasi dimulai dengan pembacaan gambar dari direktori *input* berdasarkan nama *file* yang sesuai dengan nomor set dan nomor lembar. Gambar dibaca menggunakan perintah *cv2.imread(fname, cv2.IMREAD_GRAYSCALE)*, yang mengonversinya ke format *grayscale*. Selanjutnya, binarisasi dilakukan menggunakan perintah *cv2.threshold(I, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)*. Adapun program yang digunakan pada proses binerisasi adalah sebagai berikut:

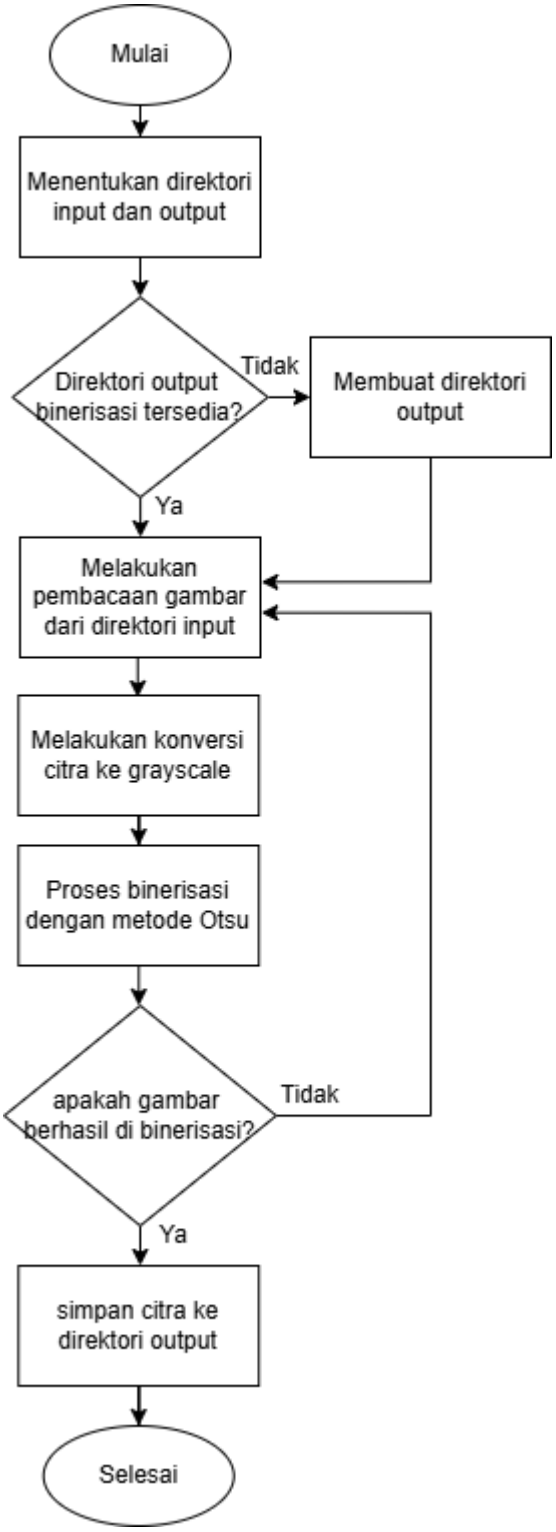
```
import cv2
import os
# Direktori input dan output
input_dir = ‘r"C:\Users\ASUS\OneDrive\Documents\ANISA RAHMA\Dataset’
output_dir =‘r"C:\Users\ASUS\OneDrive\Documents\ANISA RAHMA\Binerisasi’

# Membuat direktori output jika belum ada
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Membaca dan memproses gambar
for file_name in os.listdir(input_dir):
    if file_name.endswith(".png"):
        file_path = os.path.join(input_dir, file_name)

        # Membaca gambar dalam mode grayscale
        grayscale_image = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)

# Proses binerisasi dengan metode Otsu
_, binary_image = cv2.threshold(grayscale_image, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)

        # Menyimpan hasil binerisasi ke direktori output
```

```
    output_path = os.path.join(output_dir, file_name)
    cv2.imwrite(output_path, binary_image)
print("Proses binerisasi selesai.")
```

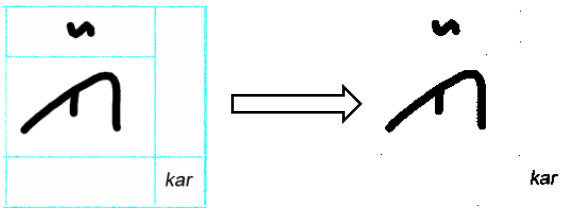
Pada tahap binarisasi ini, metode Otsu secara otomatis menentukan ambang batas terbaik (T) berdasarkan distribusi piksel dalam gambar[15]. Piksel dengan nilai di atas ambang batas (T) diubah menjadi 255 (putih), sementara piksel dengan nilai sama dengan atau di bawah ambang batas (T) diubah menjadi 0 (hitam), seperti yang ditunjukkan pada Persamaan 2.3:

$$B(x,y) = \begin{cases} 0, & jika\ I(x,y) > T \\ 255, & jika\ I(x,y) \leq T \end{cases} \quad (2.3)$$

Setelah citra menjadi *grayscale*, metode Otsu digunakan untuk menentukan ambang batas terbaik. Piksel dengan nilai intensitas di atas ambang batas berubah menjadi putih (255), dan di bawahnya menjadi hitam (0). Dalam kasus ini, warna biru pada gambar *grayscale* menghasilkan nilai intensitas tinggi, sehingga dianggap sebagai latar belakang (putih) setelah binerisasi[16]. Gambar 3.4 menggambarkan alur proses binerisasi dan Gambar 3.5 menampilkan hasil sebelum dan sesudah binarisasi.



Gambar 3. 4 Diagram Alir Proses Binerisasi



Gambar 3.5 Binerisasi Citra

c) Slicing

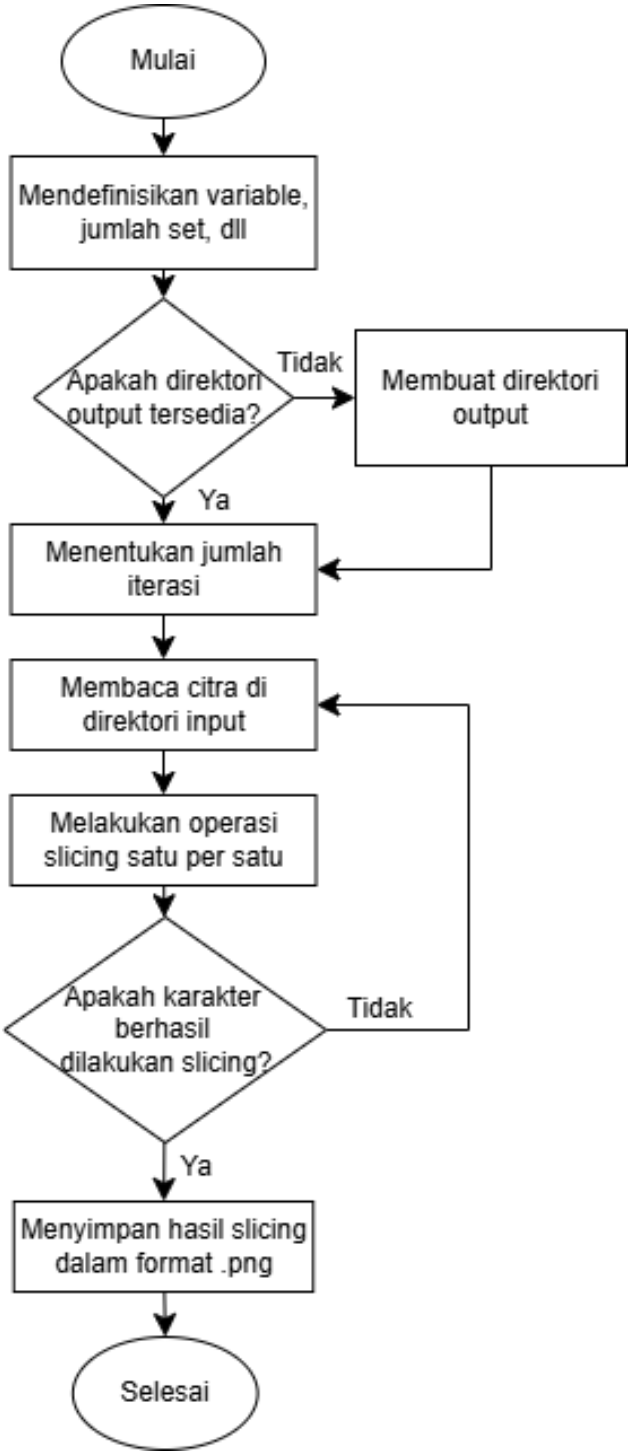
Tahapan ketiga adalah melakukan *slicing* citra dengan tujuan agar setiap aksara terpisah menjadi gambar individu. Proses ini dimulai dengan menentukan area yang berisi aksara pada citra hasil pemindaian. Setelah area terdeteksi, dilakukan pemotongan (*slicing*) secara sistematis berdasarkan letak dan ukuran aksara. Masing-masing aksara yang telah dipotong disimpan dalam *file* terpisah untuk memudahkan pengolahan lebih lanjut. Program yang digunakan pada proses *slicing* adalah sebagai berikut:

```
import cv2
import os

# Definisi variabel
aksara = [
    'ka', 'ka', 'ka', 'ka', 'ka', 'ka', 'ka', 'ka', 'ka', 'ka', 'ka', 'ka', 'ka',
    'ga', 'ga', 'ga', 'ga', 'ga', 'ga', 'ga', 'ga', 'ga', 'ga', 'ga', 'ga', 'ga',
    'nga', 'nga', 'nga', 'nga', 'nga', 'nga', 'nga', 'nga', 'nga', 'nga', 'nga', 'nga',
    'pa', 'pa', 'pa', 'pa', 'pa', 'pa', 'pa', 'pa', 'pa', 'pa', 'pa', 'pa', 'pa',
    # ... (data aksara lainnya)
    'gha', 'gha', 'gha', 'gha', 'gha', 'gha', 'gha', 'gha', 'gha', 'gha', 'gha', 'gha', 'gha',
    'nol', 'nol', 'nol', 'nol', 'nol', 'nol', 'nol', 'nol', 'nol', 'nol', 'nol', 'nol', 'nol'
]

input_dir = 'r"C:\Users\ASUS\OneDrive\Documents\ANISA RAHMA\Binerisasi'
output_dir = 'r"C:\Users\ASUS\OneDrive\Documents\ANISA RAHMA\Slicing'
number_of_set = 50
number_of_sheet = 7
```

```
number_of_char = 273
number_of_row = 4
number_of_column = 10
set_prefix = 'set'
sheet_prefix = 'sheet'
startX = 40
startY = 230
width = 191
height = 123
deltaX = 295 - startX
deltaY = 488 - startY
idx = 1

# Membuat direktori output jika belum ada
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Proses slicing citra
for m in range(1, number_of_set + 1):
    sheet_min = 1
    char_min = 1
    sheet_max = sheet_min + (number_of_sheet - 1)
    max_idx = idx + 13
    min_idx = max_idx - 13
    for n in range(sheet_min, sheet_max + 1):
        fname = f"{set_prefix}{m}{sheet_prefix}{n}.png"
        print(f"Processing file: {fname}")
        I = cv2.imread(os.path.join(input_dir, fname), cv2.IMREAD_GRAYSCALE)
        y = startY
        for i in range(number_of_row):
            x = startX
            for j in range(number_of_column):
```

```
            if char_min == number_of_char + 1:
                break
            if idx == max_idx:
                idx = min_idx
            O = I[y:y+height, x:x+width]  # Mengambil bagian citra sesuai koordinat
slicing
            fname = f"{aksara[char_min-1]}{idx}.png"
            print(f"Writing file: {fname}")
            cv2.imwrite(os.path.join(output_dir, fname), O)
            x += deltaX
            char_min += 1
            idx += 1
        y += deltaY
        sheet_min += number_of_sheet

print("Proses slicing selesai.")
```

Program diatas merupakan program untuk *slicing* citra yang diawali dengan membaca citra *input* yang sudah berbentuk citra biner dari direktori *input*. Nama *file* disusun berdasarkan nomor set dan nomor lembar, seperti set1*sheet*1.png. Citra dibaca menggunakan fungsi *cv2.imread()* dalam format *grayscale*. Koordinat awal untuk *slicing* didefinisikan dengan variabel *startX* dan *startY*, sementara ukuran potongan diatur melalui variabel *width* dan *height*. Proses *slicing* dilakukan dalam bentuk iterasi dua tingkat: tingkat pertama untuk memindahkan baris secara vertikal (dengan penambahan nilai deltaY), dan tingkat kedua untuk memindahkan kolom secara horizontal (dengan penambahan nilai deltaX). Setiap hasil potongan diberi nama *file* berdasarkan daftar aksara dan indeks karakter yang disesuaikan. Adapun alur program ditunjukkan pada Gambar 3.6



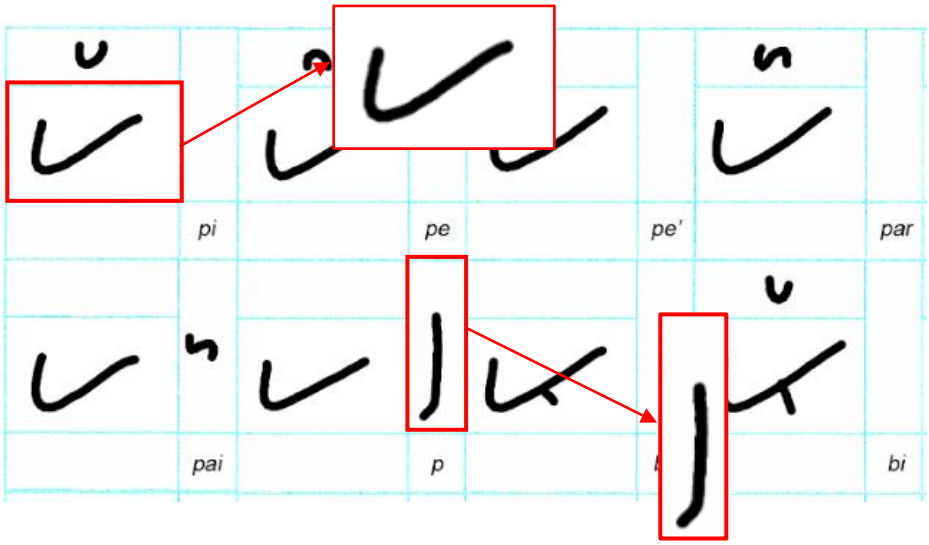
Gambar 3.6 Diagram Alir Proses *Slicing*

Proses *slicing* citra dibagi menjadi 4 proses *slicing* yaitu *slicing* induk huruf, anak huruf di atas, anak huruf di bawah, anak huruf di samping yang memiliki ukuran pemotongan berbeda. Adapun ukuran yang digunakan untuk melakukan proses *slicing* citra ditunjukan pada Tabel 3.2

Tabel 3. 2 Ukuran *Slicing* Citra

Induk Huruf	Anak Huruf Atas
startX = 148 startY = 644 width = 214 height = 118 deltaX = 453 - startX deltaY = 1070 - startY	startX = 147 startY = 567 width = 214 height = 76 deltaX = 458 - startX deltaY = 991 - startY
Anak Huruf Bawah	**Anak Huruf Depan**
startX = 147 startY = 776 width = 214 height = 76 deltaX = 458 - startX deltaY = 1190 - startY	startX = 362 startY = 569 width = 76 height = 269 deltaX = 668 - startX deltaY = 994 - startY

Hasil *slicing* ini kemudian disimpan dalam direktori *output* menggunakan fungsi *cv2.imwrite()*. Jika semua baris dan kolom telah diproses, iterasi berpindah ke *file* berikutnya hingga seluruh *file* di direktori selesai diolah. Proses ini menghasilkan potongan citra yang sesuai dengan koordinat dan nama *file* yang sudah ditentukan sebelumnya. Gambar 3.7 memperlihatkan proses *slicing* karakter aksara.



Gambar 3.7 Proses *slicing* citra

d) Inverting

Proses inversi citra dilakukan untuk membalikkan nilai piksel dalam gambar biner yang telah melalui tahap *slicing*. Proses inversi ditunjukkan dengan Persamaan 2.4.

$$g(x,y) = 255 - f(x,y) \quad (2.4)$$

Dimana g(x,y) adalah piksel setelah inverse dan f(x,y) adalah piksel citra asal[17].

Pada tahap ini, setiap piksel dengan nilai putih (255) diubah menjadi hitam (0), dan sebaliknya. Proses ini bertujuan untuk menyesuaikan format citra sehingga objek utama, yaitu aksara, memiliki warna putih, sedangkan latar belakangnya menjadi hitam. Dalam implementasi *Python*, inversi dilakukan menggunakan fungsi `cv2.bitwise_not()`, yang membalikkan semua nilai piksel dalam citra secara otomatis.

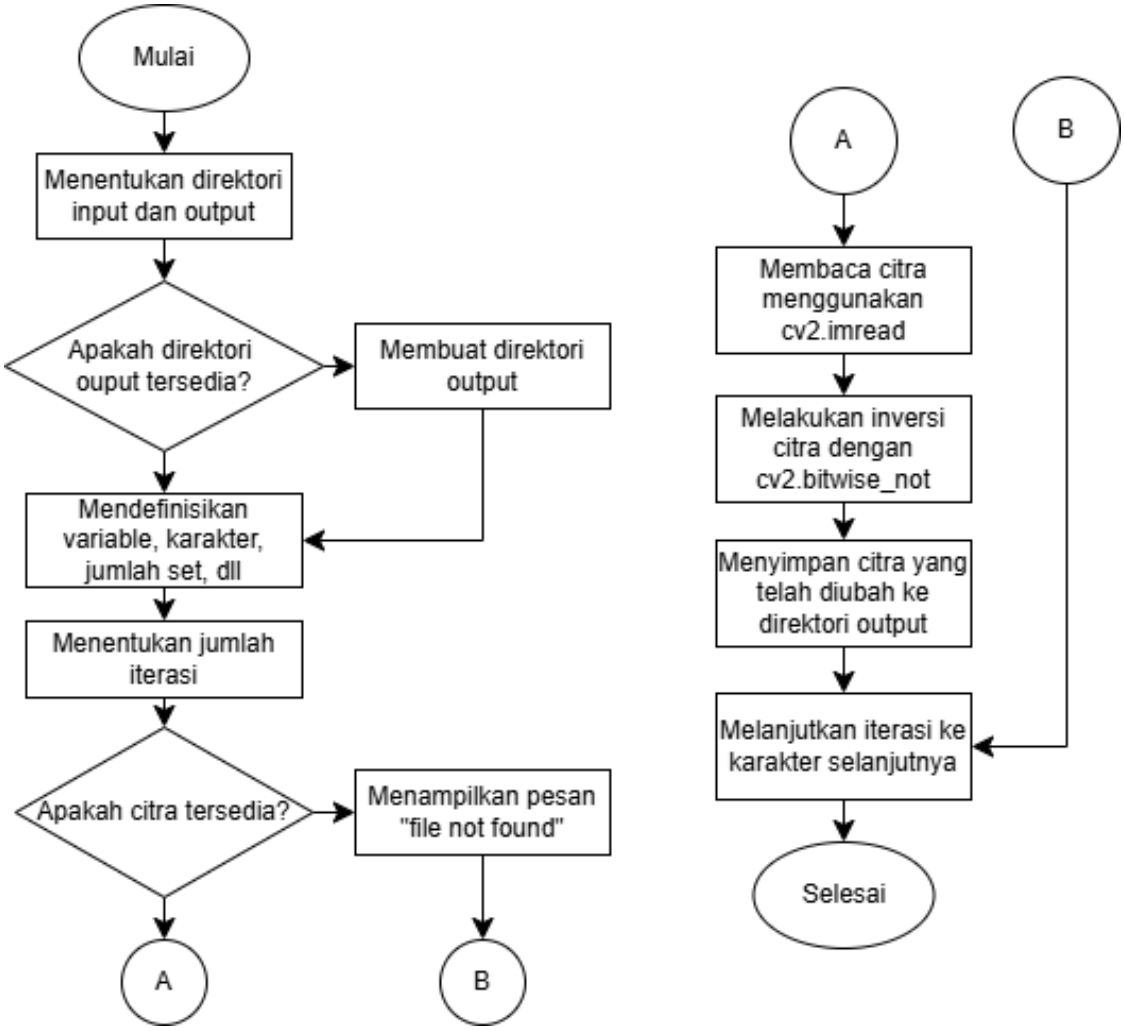
Program yang digunakan untuk proses inversi citra adalah sebagai berikut:

```
import os
import cv2
import numpy as np

# Direktori input dan output
slicing_dir = 'slicingresult'
invert_dir = 'invertresult'

# Membuat folder output jika belum ada
if not os.path.exists(cropping_dir):
    os.makedirs(cropping_dir)

# Daftar aksara
aksara = [
    'ka', 'ga', 'nga', 'pa', 'ba', 'ma', 'ta', 'da', 'na', 'ca', 'ja',
    'nya', 'ya', 'a', 'la', 'ra', 'sa', 'wa', 'ha', 'gha', 'nol'
```

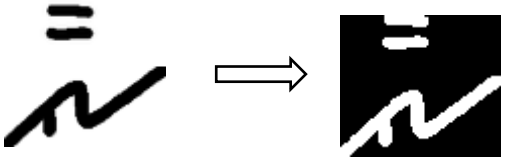
```
]

# Jumlah variasi karakter untuk semua set
number_of_char_var = 13
number_of_set = 50

# Proses inverting citra
for n in range(1, number_of_set * number_of_char_var + 1):  # Loop semua variasi karakter
    for char in aksara:  # Loop untuk setiap karakter
        fname = f"{char}{n}.png"
        input_path = os.path.join(slicing_dir, fname)
        output_path = os.path.join(cropping_dir, fname)

        if os.path.exists(input_path):  # Cek jika file input ada
            print(f"Processing: {fname}")
            # Membaca citra
            image = cv2.imread(input_path, cv2.IMREAD_GRAYSCALE)

            # Inversi citra
            inverted_image = cv2.bitwise_not(image)

            # Menyimpan hasil inversi
            cv2.imwrite(output_path, inverted_image)
        else:
            print(f"File not found: {fname}")

print("Proses inverting selesai.")
```

Setiap *file* hasil *slicing* diproses satu per satu untuk memastikan semua aksara memiliki format yang konsisten sebelum digunakan dalam analisis atau langkah pemrosesan lebih lanjut. Hasil inversi disimpan ke dalam direktori khusus untuk

memudahkan pengelolaan *file*. Adapun proses inversi dijelaskan dengan diagram alir pada Gambar 3.8 dan hasil inversi citra ditunjukkan pada Gambar 3.9



Gambar 3.8 Diagram Alir Proses Inversi



Gambar 3. 9 Hasil Inversi Citra

e) *Cropping & Resizing*

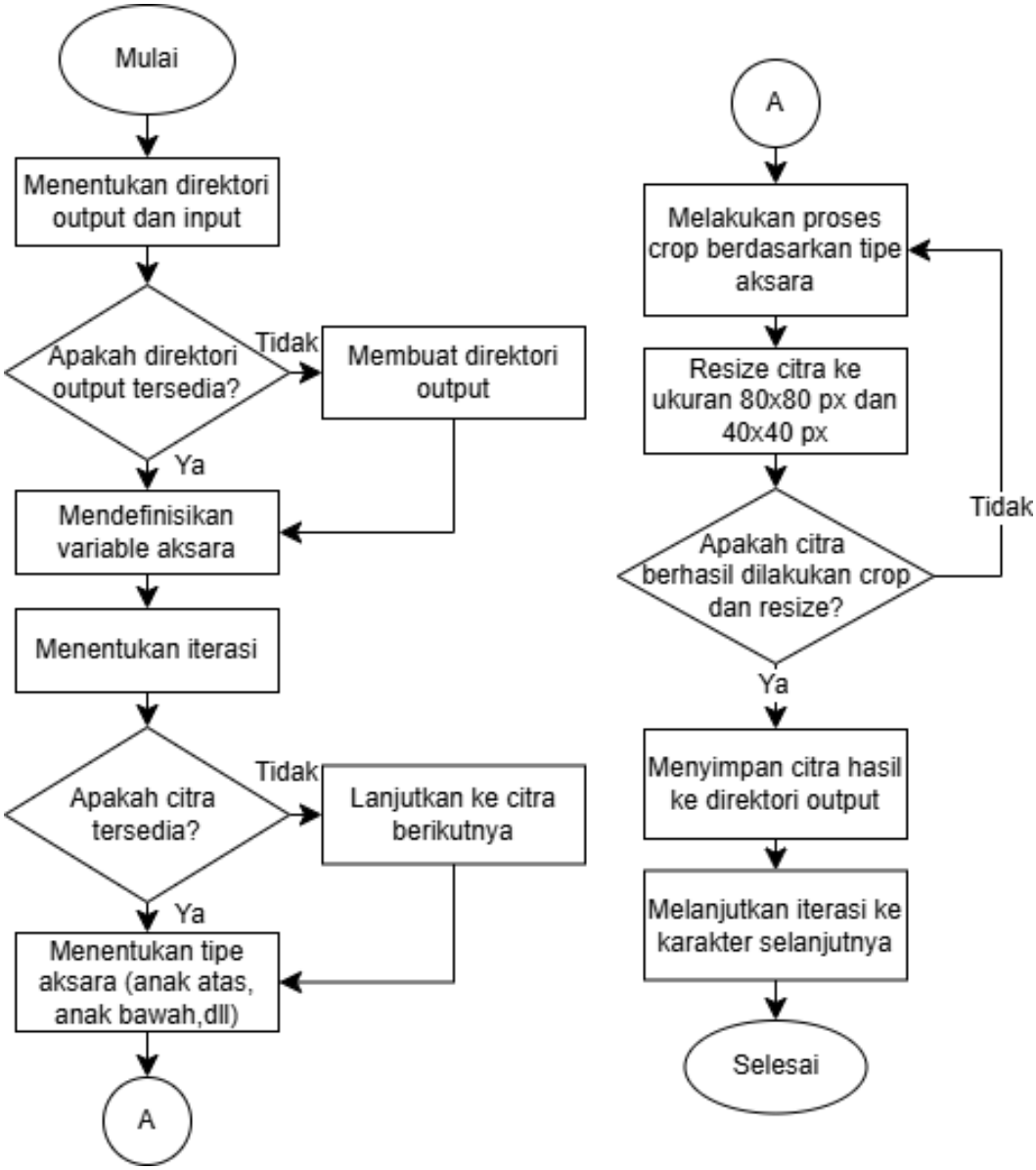
Proses *cropping* dan *resizing* citra dilakukan dengan penyesuaian berbeda untuk setiap jenis aksara, seperti anak atas, anak bawah, anak depan, dan induk huruf. Untuk aksara anak atas dan anak bawah, *cropping* dilakukan dengan menghapus 30

piksel dari sisi kiri dan kanan citra, mengurangi lebar gambar dari 214 piksel menjadi 154 piksel. Proses *cropping* dimulai dari koordinat x = 30 dan berakhir di x = 184. Setelah proses *cropping*, citra akan diubah ukuran piksel nya sesuai dengan jenis karakter menggunakan metode *resizing* dengan filter *ANTIALIAS*, yang menghasilkan gambar lebih halus. Program yang digunakan pada proses ini adalah sebagai berikut:

```
import os
from PIL import Image, ImageOps

# Direktori input dan output
inverting_dir = 'invertingresult'
cropping_dir = 'croppingresult'

# Membuat direktori output jika belum ada
if not os.path.exists(cropping_dir):
    os.makedirs(cropping_dir)

# Daftar nama karakter
aksara = [
    'ka', 'ga', 'nga', 'pa', 'ba', 'ma', 'ta', 'da', 'na', 'ca', 'ja',
    'nya', 'ya', 'a', 'la', 'ra', 'sa', 'wa', 'ha', 'gha', 'nol'
]

# Proses cropping dan resizing
def process_image(input_path, output_path, aksara_type):
    print(f"Processing: {input_path}")

    I = Image.open(input_path).convert('1')

    if aksara_type in ['anak_atas', 'anak_bawah']:
        # Cropping untuk anak atas dan anak bawah
```

```
        cropped = I.crop((30, 0, 184, I.height))  # Crop dari x=30 ke x=184 (154 px width)
    else:
        # Default cropping (tidak ada aturan spesifik disebutkan)
        bbox = I.getbbox()
        cropped = I.crop(bbox) if bbox else I

    # Resizing
    resized = cropped.resize((80, 70), Image.ANTIALIAS)

    # Simpan hasil
    print(f"Writing file: {output_path}")
    resized.save(output_path)

# Loop untuk semua variasi karakter
numberofcharvar = 13  # Jumlah variasi untuk setiap karakter
numberofset = 50  # Jumlah set data

for n in range(1, numberofcharvar * numberofset + 1):  # Loop untuk setiap variasi
    for m, char_name in enumerate(aksara):  # Loop untuk setiap karakter
        fname = f"{char_name}{n}.png"
        input_path = os.path.join(inverting_dir, fname)

        if not os.path.exists(input_path):  # Skip jika file tidak ditemukan
            continue

        # Tentukan tipe aksara (contoh: anak atas, anak bawah, atau default)
        aksara_type = 'anak_atas' if 'ya' in char_name else 'anak_bawah' if 'wa' in char_name else 'default'

        output_path = os.path.join(cropping_dir, fname)
        process_image(input_path, output_path, aksara_type)
```

Proses *cropping* dan *resizing* dijelaskan pada Gambar 3.10



Gambar 3.10 Diagram Alir Proses *Cropping* dan *Resizing*

Pada induk huruf, jika aksara bukan yang terakhir, *cropping* dilakukan dengan mencari batas-batas minimum dan maksimum dari piksel putih (yang mewakili aksara) menggunakan koordinat dari *np.where.* Citra kemudian dipotong mengikuti batas-batas tersebut, memastikan aksara terisolasi secara presisi. Jika tidak ada piksel putih yang ditemukan, citra asli tetap dipertahankan tanpa modifikasi. Setelah *cropping*, citra akan di perkecil untuk keseragaman format sebelum digunakan dalam penyatuan gambar antara induk huruf dengan anak huruf. Perbandingan citra sebelum dan sesudah proses *crop* dan *resize* ditunjukkan pada Gambar 3.11

80x80 piksel	40x40 piksel
	

f) Converting

Tahap akhir dari *pre-processing* adalah mengubah format gambar dari PNG menjadi *numpy array* dengan nilai piksel biner, yaitu 0 dan 1. Proses ini dimulai dengan membaca gambar dalam format PNG dan mengonversinya menjadi *array numpy*. Gambar dibaca dalam mode *grayscale* untuk memastikan setiap piksel hanya memiliki satu nilai intensitas. Pada tahap ini, citra yang telah melalui proses binerisasi awal dan *cropping* dikonversi menjadi *numpy array* biner. Proses ini memastikan bahwa citra akhir fokus pada bentuk dan kontur aksara tanpa gangguan dari variasi warna atau intensitas yang mungkin ada setelah proses *cropping* dan *resizing*.

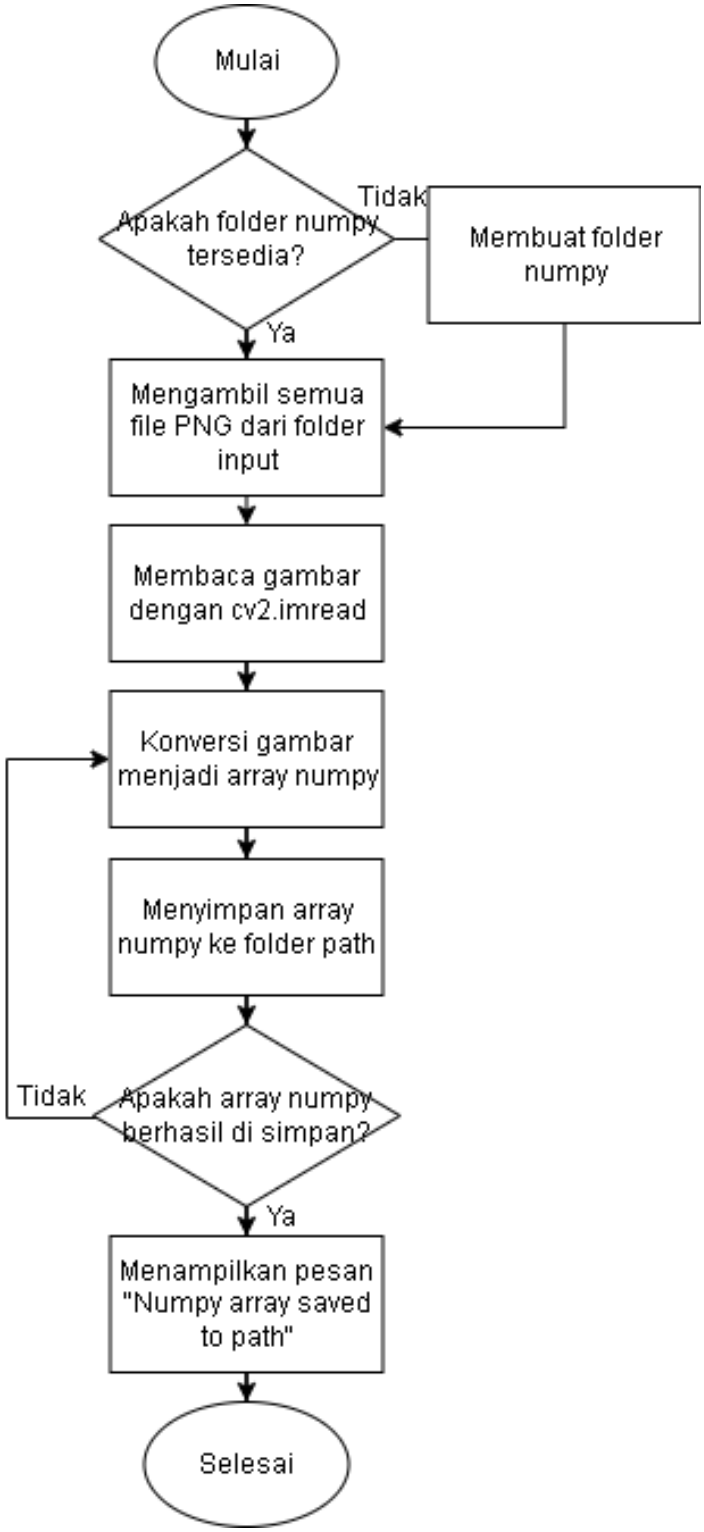
Proses pada kode ini dimulai dengan memastikan folder tujuan untuk menyimpan *file numpy (numpy_folder)* ada. Jika folder belum ada, maka akan dibuat menggunakan fungsi *os.makedirs()*. Setelah itu, program membaca daftar semua *file* dengan ekstensi .png yang ada di folder sumber *(folder_path)* menggunakan fungsi *os.listdir()* dan menyaring *file* berdasarkan ekstensinya. Jika tidak ditemukan *file* PNG, program akan mencetak pesan bahwa folder kosong dan langsung menghentikan proses. Selanjutnya, program memproses setiap *file* PNG dalam folder. Untuk setiap *file*, program membuat path lengkap *file* menggunakan *os.path.join()* agar dapat diakses. Gambar PNG kemudian dibuka menggunakan *Library Pillow* dengan fungsi *Image.open(),* tetapi tidak ada perubahan atau konversi lebih lanjut terhadap gambar tersebut. Gambar tersebut langsung diubah menjadi *array numpy* menggunakan fungsi *np.array(),* sehingga data gambar dalam format numerik siap diproses.

Setelah data gambar dalam format *array numpy* diperoleh, program membuat nama *file* keluaran dengan mengganti ekstensi *file* PNG menjadi .npy menggunakan *os.path.splitext(). File numpy* kemudian disimpan di folder tujuan dengan

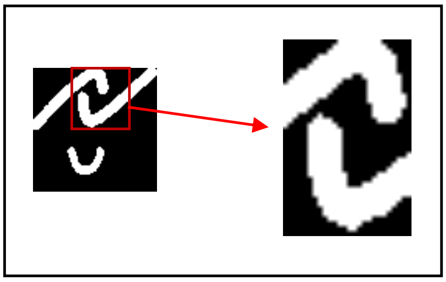
menggunakan fungsi *np.save(),* dan program mencetak pesan keberhasilan untuk setiap *file* yang disimpan. Proses ini diulangi untuk semua *file* PNG yang ada di folder sumber. Adapun program yang digunakan untuk proses convert PNG ke NPY adalah sebagai berikut:

```
def convert_images_to_numpy(folder_path, numpy_folder):
    # Memastikan folder numpy ada
    if not os.path.exists(numpy_folder):
        os.makedirs(numpy_folder)
     # Mendapatkan daftar semua file PNG dalam folder
    png_files = [f for f in os.listdir(folder_path) if f.endswith('.png')]

    # Memproses setiap gambar PNG dalam folder
    for file_name in png_files:
        file_path = os.path.join(folder_path, file_name)

        # Mengonversi gambar menjadi array numpy
        img_array = np.array(img)
            # Menyimpan array numpy ke file .npy di folder numpy_folder
        numpy_file_name = os.path.splitext(file_name)[0] + '.npy'
        numpy_file_path = os.path.join(numpy_folder, numpy_file_name)
        np.save(numpy_file_path, img_array)
        print(f"Numpy array saved to {numpy_file_path}")

# Path ke folder yang berisi file PNG
folder_path = 'D:/ANISA RAHMA/resize data training'

# Folder untuk menyimpan file numpy array
numpy_folder = 'D:/ANISA RAHMA/trainresize npy'

# Memproses semua gambar dalam folder
convert_images_to_numpy(folder_path, numpy_folder)
```

Program yang tertera diatas ditunjukkan pada Gambar 3.12 dan hasil dari proses *converting* ditunjukkan pada Gambar 3.13



Gambar 3.11 Diagram Alir Proses *Converting*



0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0
1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0
0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0

Gambar 3.12 Contoh *Converting* PNG ke NPY

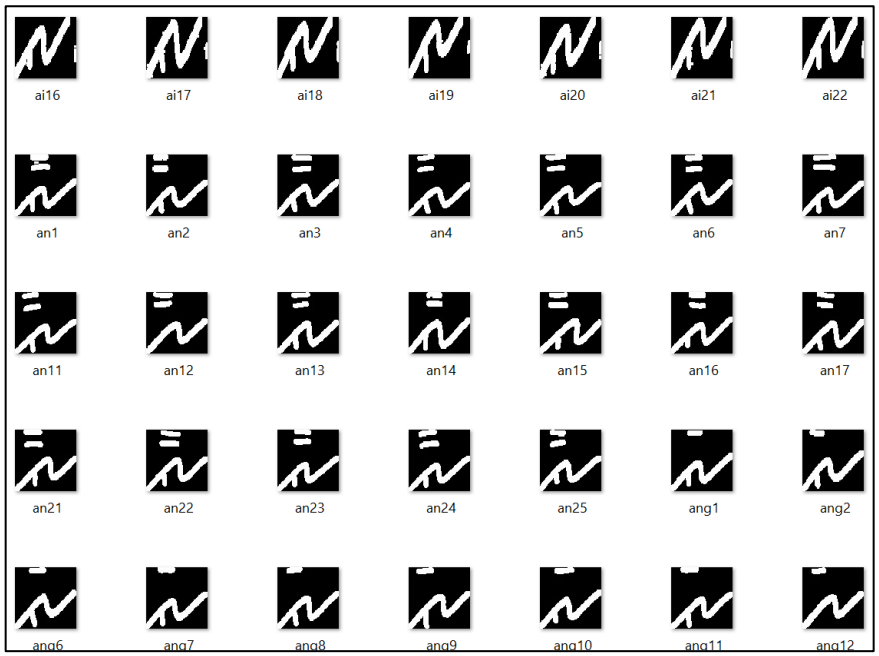
3.4.2 *Dataset*

Penelitian ini menggunakan *dataset* berupa citra aksara lampung yang dikumpulkan melalui angket. Pada penelitian ini menggunakan *dataset* yang diperoleh dari citra aksara sebanyak 6.500 karakter aksara Lampung dengan pembagian yang ditunjukkan pada tabel 3.3

Tabel 3.3 Pembagian Dataset

NO	Karakter Aksara		Data *Template*		Data Uji	
	Aksara	Latin	Variasi Anak Huruf	Jumlah Angket	Variasi Anak Huruf	Jumlah Angket
1		ka	13	25	13	5
2		ga	13	25	13	5
3		nga	13	25	13	5
4		pa	13	25	13	5
5		ba	13	25	13	5
6		ma	13	25	13	5
7		ta	13	25	13	5
8		da	13	25	13	5
9		na	13	25	13	5
10		ca	1	25	13	5
11		ja	13	25	13	5
12		nya	13	25	13	5
13		ya	13	25	13	5
14		a	13	25	13	5
15		la	13	25	13	5
16		ra	13	25	13	5
17		sa	13	25	13	5
18		wa	13	25	13	5
19		ha	13	25	13	5
20		gha	13	25	13	5
Total			6.500		1.300	

3.4.3 Data *Template*



Gambar 3.13 Data *Template*

Penelitian ini menggunakan satu set data *template* yang terdiri dari 6.500 karakter aksara Lampung, yang diperoleh dari 20 angket tulisan tangan. Dataset ini kemudian dibagi menjadi tiga set data *template* yang digunakan dalam proses pengujian untuk mengevaluasi performa model terhadap jumlah data yang berbeda, yaitu:

- Set 1: 6.500 karakter (keseluruhan dataset)
- Set 2: 3.900 karakter (subset dari dataset utama)
- Set 3: 1.300 karakter (subset yang lebih kecil)

Untuk memastikan model dapat beradaptasi terhadap variasi ukuran citra, setiap pengujian dilakukan dalam dua ukuran berbeda, yaitu 80×80 piksel dan 40×40 piksel. Proses konversi dilakukan dengan mengubah setiap karakter dalam dataset menjadi representasi dalam format *array numpy*, sehingga dapat diolah lebih lanjut dalam tahap pe*template*an dan pengujian model.

Dengan adanya tiga set data *template* dan dua ukuran citra, maka dilakukan total enam kali pengujian pada setiap metode untuk mengamati pengaruh jumlah data dan ukuran citra terhadap akurasi. Pengujian ini bertujuan untuk menganalisis

bagaimana model dapat mengenali pola karakter aksara Lampung dalam berbagai kondisi, serta menentukan konfigurasi dataset yang memberikan hasil terbaik.

3.4.4 Data Uji

Dalam penelitian ini, data uji digunakan untuk menguji dan memvalidasi hasil identifikasi yang diperoleh dari data *template*. Data uji terdiri dari citra baru yang belum pernah digunakan sebelumnya dalam proses pe*template*an, sehingga dapat memberikan evaluasi yang objektif terhadap keakuratan metode dalam mengenali karakter aksara Lampung. Dataset uji dalam penelitian ini terdiri dari 1.300 karakter, yang dibuat dalam dua ukuran citra berbeda, yaitu:

- 80×80 piksel
- 40×40 piksel

Proses pengujian dilakukan dengan mencocokkan ukuran data uji dengan ukuran data *template*. Artinya, model yang di*template* menggunakan data berukuran 80×80 piksel akan diuji dengan data uji yang memiliki ukuran yang sama (80×80 piksel). Begitu pula dengan model yang di*template* menggunakan data berukuran 40×40 piksel, pengujiannya akan dilakukan dengan data uji 40×40 piksel.

Pendekatan ini diterapkan karena metode yang digunakan dalam penelitian ini, yaitu *Sum of Absolute Differences* (SAD) dan *Sum of Squared Difference*s (SSD), hanya dapat melakukan perhitungan pada citra dengan ukuran yang sama[11][18]. Oleh karena itu, memastikan kesesuaian ukuran antara data *template* dan data uji menjadi aspek penting dalam desain eksperimen untuk memperoleh hasil yang valid dan akurat.

Dengan metode ini, penelitian dapat mengevaluasi seberapa baik model dalam mengenali karakter aksara Lampung dalam berbagai skenario, serta memahami bagaimana ukuran citra memengaruhi performa metode identifikasi yang digunakan. Adapun contoh citra yang digunakan sebagai data uji ditunjukkan pada Gambar 3.15.

Gambar 3.14 Data Uji

Secara teknis, proses pengambilan data uji sama dengan saat pengambilan data *template*, yaitu melalui tahapan *pre-processing* terlebih dahulu. Tahapan tersebut mencakup pengambilan gambar menggunakan perangkat pemindai, kemudian dilakukan *cropping* dan *resizing* untuk menyesuaikan ukuran citra, serta mengubah citra berwarna menjadi citra biner. Data yang siap digunakan sebagai data uji adalah data yang memiliki ukuran piksel yang sama dengan data *template*, serta telah melalui proses konversi menjadi citra biner. Selanjutnya, data uji ini diuji dengan metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dalam program yang telah dikembangkan menggunakan *Python*. Pengujian dilakukan secara bertahap pada masing-masing karakter aksara Lampung, dimana identifikasi dilakukan per karakter.

3.5 *Sum of Absolute Differencess* (SAD) dan *Sum of Squared Differencess* (SSD)

Dalam penelitian ini, *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) digunakan sebagai dua metrik utama untuk mengukur kemiripan antara blok-blok citra uji dengan blok-blok citra referensi[19]. Kedua metrik ini berfungsi untuk mengevaluasi seberapa dekat karakteristik citra uji dengan template yang telah ditentukan sebelumnya. Baik SAD maupun SSD memiliki peran penting dalam proses identifikasi karakter aksara Lampung, yang terdiri dari induk huruf dan variasi anak huruf.

SAD adalah metrik yang menghitung jumlah perbedaan mutlak antara nilai piksel pada posisi yang sesuai antara blok citra uji dan blok referensi. Metrik ini menilai seberapa besar perbedaan antara dua blok citra dengan menjumlahkan perbedaan absolut antara setiap pasangan piksel yang bersesuaian dalam kedua blok. SAD memberikan penekanan yang sama terhadap semua perbedaan antara piksel, sehingga sangat sensitif terhadap perubahan kecil pada setiap piksel. Hasil dari perhitungan SAD yang lebih rendah menunjukkan kemiripan yang lebih tinggi antara blok citra uji dan blok referensi, karena perbedaan antara kedua blok lebih kecil.

Berbeda dengan SAD, SSD menghitung perbedaan kuadrat antara nilai piksel yang bersesuaian dalam dua blok citra. SSD memberikan bobot yang lebih besar terhadap perbedaan yang lebih besar antara nilai piksel, sehingga perbedaan yang lebih besar akan memberikan kontribusi yang lebih signifikan terhadap nilai total SSD. Dengan demikian, SSD lebih sensitif terhadap perbedaan besar antara blok citra dibandingkan SAD. Hasil perhitungan SSD yang lebih rendah juga menunjukkan kemiripan yang lebih tinggi, karena semakin kecil nilai perbedaan kuadrat antara kedua blok citra.

Pada setiap pengujian, baik untuk SAD maupun SSD, langkah pertama adalah mengubah citra uji menjadi format yang dapat diproses, yaitu dengan mengubah citra menjadi *array numpy* dan melakukan normalisasi piksel ke rentang 0 hingga 1. Kemudian, kedua metrik SAD dan SSD dihitung untuk mengukur kemiripan antara blok citra uji dengan blok citra referensi yang sudah ada dalam database.

Sebagai ilustrasi, berikut adalah contoh perhitungan SAD dan SSD untuk citra biner dalam ukuran 3×3 piksel menggunakan Persamaan 2.1 dan 2.2

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Tabel 3.4 Contoh Perhitungan SAD dan SSD

$SAD = \sum_i \sum_j \lvert A(i,j) - B(i,j) \rvert$	$SSD = \sum_i \sum_j \big(A(i,j) - B(i,j)\big)^2$
$= \begin{bmatrix} \lvert 1-0 \rvert & \lvert 0-1 \rvert & \lvert 1-0 \rvert \\ \lvert 0-0 \rvert & \lvert 1-1 \rvert & \lvert 0-1 \rvert \\ \lvert 1-1 \rvert & \lvert 0-0 \rvert & \lvert 1-0 \rvert \end{bmatrix}$	$= \begin{bmatrix} \lvert 1-0 \rvert & \lvert 0-1 \rvert & \lvert 1-0 \rvert \\ \lvert 0-0 \rvert & \lvert 1-1 \rvert & \lvert 0-1 \rvert \\ \lvert 1-1 \rvert & \lvert 0-0 \rvert & \lvert 1-0 \rvert \end{bmatrix}$
$= \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$	$= \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$
$= 0+1+1+0+0+1+0+0+1$	$= 0+1+1+0+0+1+0+0+1$
$= 4$	$= 4$

Dari contoh perhitungan ini, terlihat bahwa pada citra biner, nilai SAD dan SSD selalu sama. Hal ini terjadi karena dalam biner hanya ada dua kemungkinan nilai piksel (0 dan 1), sehingga selisih absolut dan kuadrat dari selisih akan selalu memiliki hasil yang sama untuk setiap perbedaan piksel. Oleh karena itu, baik SAD maupun SSD memberikan hasil identik pada citra biner. Adapun proses perhitungan SAD dan SSD secara menyeluruh pada penelitian ini ditunjukkan dengan listing program dibawah ini.

```
import os
import numpy as np
import pandas as pd
import time

# Folder testing dan training
testing_folder = "D:/ANISA RAHMA/NPY DATA BARU/TESTING/resize induk huruf"
training_folder = "D:/ANISA RAHMA/NPY DATA BARU/TRAINING/resize induk huruf"
```

```
# Ambil daftar file .npy dari folder
testing_files = [f for f in os.listdir(testing_folder) if f.endswith(".npy")]
training_files = [f for f in os.listdir(training_folder) if f.endswith(".npy")]

# Validasi keberadaan file
if not testing_files:
    raise FileNotFoundError(f"Folder testing kosong. Pastikan terdapat file .npy di folder {testing_folder}")
if not training_files:
    raise FileNotFoundError(f"Folder training kosong. Pastikan terdapat file .npy di folder {training_folder}")

# Tentukan nama Sheet berdasarkan nama folder testing
sheet_name = os.path.basename(testing_folder).replace(" ", "_")

# Path output file Excel
output_file = "D:/ANISA RAHMA/SAD_Test_Python.xlsx"

# Cek apakah sheet sudah ada, jika ya, buat nama baru
with pd.ExcelWriter(output_file, mode="a", engine="openpyxl", if_sheet_exists="overlay") as writer:
    existing_sheets = writer.book.sheetnames if writer.book else []

    if sheet_name in existing_sheets:
        count = 2
        while f"{sheet_name}_{count}" in existing_sheets:
            count += 1
        sheet_name = f"{sheet_name}_{count}"

# Parameter batch size
batch_size = 10  # Sesuaikan batch size sesuai kemampuan laptop
num_batches = (len(testing_files) + batch_size - 1) // batch_size  # Round up

# Tulis header ke Excel
```

```
headers = ["Testing File", "Best match", "SAD Value", "Time (seconds)"]
df_headers = pd.DataFrame([headers])
df_headers.to_excel(output_file, sheet_name=sheet_name, index=False, header=False)

# Loop untuk setiap batch
all_results = []

for batch_idx in range(num_batches):
    # Tentukan indeks file dalam batch ini
    start_idx = batch_idx * batch_size
    end_idx = min((batch_idx + 1) * batch_size, len(testing_files))

    # List hasil batch
    batch_results = []

    # Loop untuk setiap file testing dalam batch
    for i in range(start_idx, end_idx):
        testing_file_path = os.path.join(testing_folder, testing_files[i])
        testing_data = np.load(testing_file_path)

        print(f"Sedang memproses file testing: {testing_files[i]}")

        # Inisialisasi SAD minimum
        min_sad = float("inf")
        best_match = ""
        best_time = 0

        # Loop untuk setiap file training
        for training_file in training_files:
            training_file_path = os.path.join(training_folder, training_file)
            training_data = np.load(training_file_path)
```

```
            # Pastikan ukuran array sama
            if testing_data.shape == training_data.shape:
                # Hitung SAD dan waktu komputasi
                start_time = time.time()
                sad_value = np.sum(np.abs(testing_data - training_data))
                elapsed_time = time.time() - start_time

                # Update jika SAD lebih kecil
                if sad_value < min_sad:
                    min_sad = sad_value
                    best_match = training_file
                    best_time = elapsed_time
            else:
                print(f"Warning: Ukuran array berbeda untuk {testing_files[i]} dan {training_file}, dilewati.")

        # Simpan hasil ke batchResults
        batch_results.append([testing_files[i], best_match, min_sad, best_time])
        print(f"Hasil sementara: Best match = {best_match}, SAD = {min_sad}, Waktu = {best_time:.6f} detik")

    # Simpan batch hasil ke all_results
    all_results.extend(batch_results)

# Tulis hasil batch ke Excel
df_results = pd.DataFrame(all_results, columns=headers)
with pd.ExcelWriter(output_file, mode="a", engine="openpyxl", if_sheet_exists="overlay") as writer:
    df_results.to_excel(writer, sheet_name=sheet_name, startrow=1, index=False, header=False)

print(f"Proses selesai, hasil SAD disimpan di Excel pada sheet: {sheet_name}")
```

Program diatas merupakan proses perhitungan *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) yang dimulai dengan mempersiapkan sistem untuk membandingkan *file testing* dan *training* dalam format *.npy*. Tahapan pertama adalah menentukan folder *input* dan *output*, yaitu folder untuk *file testing* (*testing*FolderNpy) dan *training* (*training*FolderNpy) yang berisi *file* .npy serta folder *output* (*testing*FolderNpy dan *training*FolderNpy) untuk menyimpan hasil perhitungan dalam format .npy

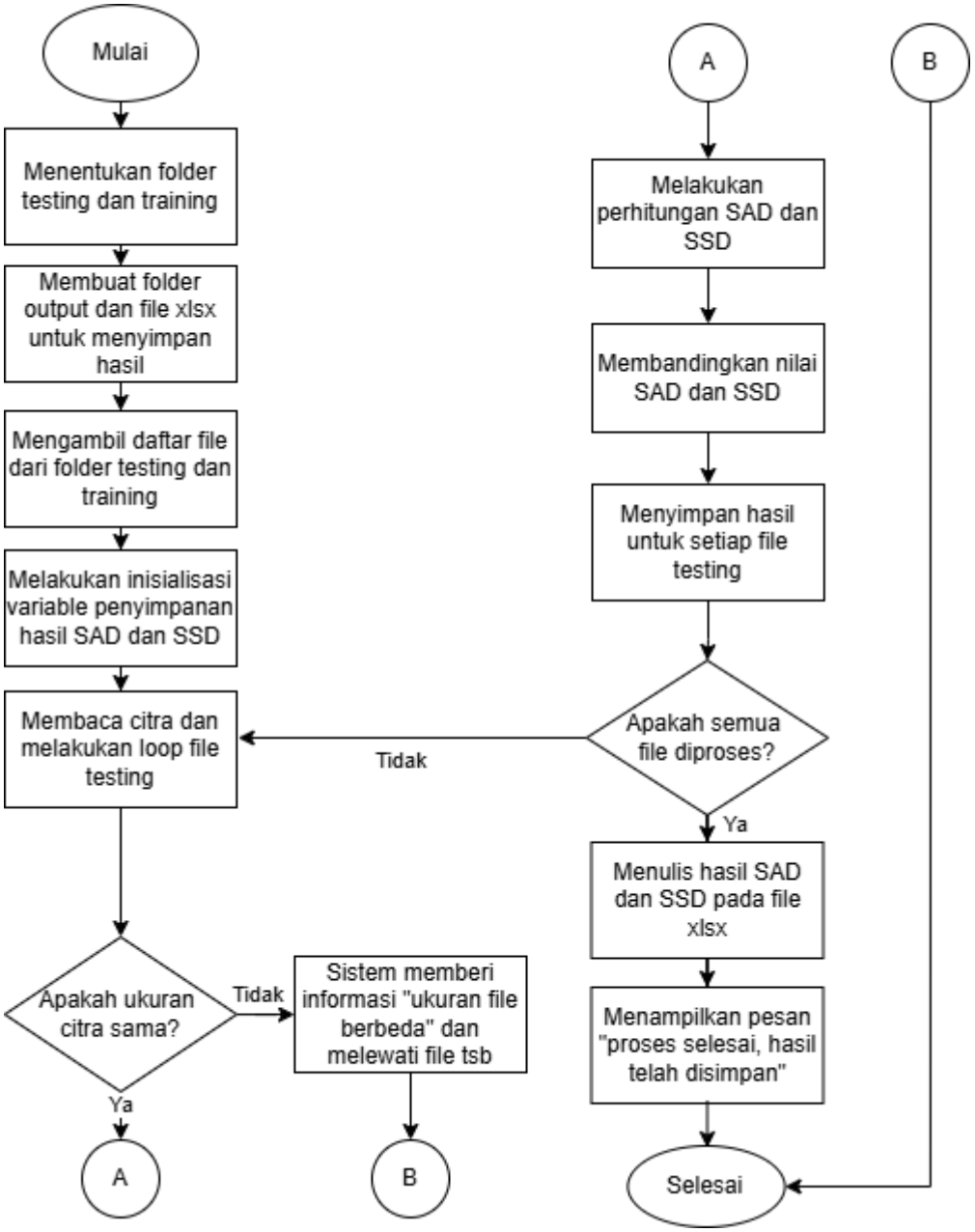
Sistem akan memeriksa ketersediaan folder *output*. Jika folder belum ada, sistem secara otomatis membuat folder baru agar hasil dapat disimpan. Setelah itu, sistem mengambil daftar *file* .npy dari folder *testing* dan *training*. Jika *file* tidak ditemukan, proses akan dihentikan dengan pesan *error*. Kemudian, variabel untuk menyimpan hasil perhitungan SAD dan SSD diinisialisasi dalam bentuk *cell array*, mencakup informasi nama *file testing*, nama *file best match*, nilai SAD atau SSD terkecil, dan waktu komputasi.

Sistem memulai proses *loop* untuk setiap *file* di folder *testing*. Dalam setiap iterasi, *file testing* dibaca menggunakan fungsi readNPY, dan variabel untuk nilai minimum SAD, SSD, *best match*, serta waktu komputasi diinisialisasi. Selanjutnya, dilakukan *loop* untuk semua *file* di folder *training*, di mana setiap *file training* dibaca menggunakan fungsi yang sama.

Sebelum menghitung SAD dan SSD, sistem memastikan bahwa ukuran *file testing* dan *training* sama. Jika ukuran *file* berbeda, sistem akan memberikan peringatan dan melewati *file* tersebut. Perhitungan SAD dilakukan dengan menjumlahkan perbedaan absolut elemen-elemen data antara *file testing* dan *training*, sementara perhitungan SSD dilakukan dengan menjumlahkan kuadrat dari perbedaan elemen-elemen data tersebut. Waktu komputasi untuk masing-masing perhitungan dicatat menggunakan fungsi tic dan toc.

Jika nilai SAD atau SSD yang baru dihitung lebih kecil dari nilai minimal sebelumnya, sistem memperbarui nilai minimal tersebut, nama *file best match*, dan waktu komputasi. Setelah semua *file training* dibandingkan dengan satu *file testing*, hasil terbaik (*best match*, nilai minimal, dan waktu komputasi) disimpan dalam *array* hasil (resultsSAD untuk SAD dan *results*SSD untuk SSD).

Proses ini berulang untuk semua *file testing* hingga selesai. Setelah semua *file* diproses, hasil SAD dan SSD ditulis ke dalam *file* Excel, masing-masing dalam *sheet* terpisah. *Sheet* pertama berisi informasi SAD, sedangkan *sheet* kedua berisi informasi SSD. Sistem menampilkan pesan bahwa proses telah selesai dan hasil telah disimpan. Akhirnya, proses dihentikan setelah semua langkah selesai dijalankan. Adapun proses program diatas dapat dijelaskan dengan diagram alir yang ditunjukkan pada Gambar 3.17



Gambar 3.15 Diagram Alir Proses Perhitungan SAD dan SSD

Akurasi pengujian dihitung berdasarkan jumlah karakter yang dikenali dengan benar dalam pengujian menggunakan SAD dan SSD. Jumlah pengujian dihitung berdasarkan jumlah data uji yang telah diproses[20]. Perhitungan akurasi dilakukan dengan persamaan:

$$\%\ Akurasi = \frac{Jumlah\ benar}{Jumlah\ pengujian} \times 100\% \quad (2.6)$$

Selain itu, tingkat kesalahan dihitung untuk mengukur seberapa besar persentase kesalahan dalam identifikasi karakter. Perhitungan kesalahan dilakukan dengan rumus:

$$\%\ Kesalahan = \frac{Jumlah\ kesalahan}{Jumlah\ pengujian} \times 100\% \quad (2.7)$$

Dengan menggunakan SAD dan SSD, tingkat akurasi dan kesalahan pada setiap pengujian dapat dihitung dan dibandingkan untuk setiap ukuran citra dan set data *template* yang digunakan. Melalui pengujian yang dilakukan sebanyak 6 kali dengan dua ukuran citra yang berbeda, hasilnya dapat memberikan gambaran yang lebih jelas mengenai performa SAD dan SSD dalam mengenali karakter aksara Lampung.

Dengan membandingkan tingkat akurasi dan kesalahan antara kedua metrik, serta dengan ukuran citra yang berbeda, dapat dianalisis metode mana yang memberikan hasil yang lebih baik dalam pengenalan karakter aksara Lampung. Selain itu, pengujian ini juga akan menunjukkan bagaimana perubahan ukuran citra memengaruhi kemampuan sistem dalam mengenali karakter, serta bagaimana jumlah data *template* yang berbeda dapat mempengaruhi hasil identifikasi karakter.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan mengenai perbandingan tingkat akurasi metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) dalam identifikasi pola aksara Lampung, dapat disimpulkan bahwa:

1. Metode *Sum of Absolute Differences* (SAD) dan *Sum of Squared Differences* (SSD) menunjukkan tingkat akurasi yang identik dalam pengenalan karakter aksara Lampung. Akurasi meningkat seiring bertambahnya jumlah dataset, dari 77,46%–77,92% pada 1.300 karakter menjadi 87,31%–90,54% pada 6.500 karakter, dengan akurasi tertinggi dicapai pada ukuran citra 80x80 piksel.
2. Persentase kesalahan menurun seiring dengan bertambahnya jumlah dataset. Pada dataset 1.300 karakter, kesalahan mencapai 22,54% (40x40 px) dan 22,08% (80x80 px), sedangkan pada dataset 6.500 karakter turun menjadi 12,69% (40x40 px) dan 9,46% (80x80 px). Kesalahan umumnya terjadi pada karakter dengan bentuk yang mirip.
3. Dengan akurasi yang setara tetapi waktu komputasi yang jauh lebih tinggi, metode SAD lebih direkomendasikan dibandingkan SSD untuk pengenalan karakter aksara Lampung. Hal ini disebabkan oleh perbedaan bentuk formula antara kedua metode, di mana SAD menggunakan lebih sedikit operasi dibandingkan SSD. Selain itu, komputer atau perangkat memiliki fungsi **np.abs** yang telah tersedia untuk menghitung nilai absolut secara langsung, sedangkan SSD tidak memiliki fungsi khusus sehingga perhitungannya harus dilakukan dengan menjumlahkan selisih terlebih dahulu, lalu menguadratkan hasilnya. Kompleksitas tambahan ini membuat metode SSD lebih lambat dalam

komputasi. Oleh karena itu, metode SAD lebih efisien dalam hal waktu pemrosesan, terutama untuk aplikasi yang membutuhkan performa tinggi.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa hal yang dapat diperhatikan untuk pengembangan lebih lanjut dalam identifikasi pola aksara Lampung. Berikut beberapa saran yang dapat dipertimbangkan:

1. Penggunaan citra berukuran lebih besar dan jumlah dataset yang lebih banyak disarankan untuk meningkatkan akurasi pengenalan pola aksara Lampung. Ukuran gambar yang lebih besar memberikan lebih banyak detail, sementara jumlah dataset yang lebih banyak membantu model mengenali pola dengan lebih baik, sehingga meningkatkan akurasi pengenalan.
2. Metode SAD lebih direkomendasikan dibandingkan SSD untuk pengenalan pola aksara Lampung, terutama dalam aplikasi yang membutuhkan efisiensi waktu komputasi. SAD terbukti memiliki waktu komputasi yang lebih cepat dibandingkan SSD tanpa kehilangan akurasi yang signifikan, sehingga lebih sesuai untuk implementasi dalam sistem real-time.
3. Penelitian selanjutnya dapat mengeksplorasi kombinasi metode pengenalan pola lain atau pendekatan tambahan, seperti *deep learning* untuk meningkatkan akurasi. Hal ini penting untuk mengatasi kesalahan pengenalan pada karakter dengan bentuk yang mirip, yang menjadi tantangan pada metode SAD dan SSD.

DAFTAR PUSTAKA

[1] S. Istiqphara, A. N. Faida, and A. U. Darajat, "Pengenalan Aksara Kaganga Lampung dengan Menggunakan Metode K-Nearest Neighbour (K-NN)," *ELECTRON J. Ilm. Tek. Elektro*, vol. 4, no. 1, pp. 15–20, 2023, doi: 10.33019/electron.v4i1.39.

[2] E. Hara, "Sistem Pengenalan Tulisan Tangan Aksara Lampung Dengan Metode Deteksi Tepi (Canny) Berbasis Jaringan Syaraf Tiruan Backpropagation," *Electr. J. Rekayasa dan Teknol. Elektro*, vol. 10, no. 3, pp. 1–86, 2016.

[3] A. Aryantio, "Pengenalan Aksara Lampung Menggunakan Jaringan Syaraf Tiruan," pp. 1–5.

[4] A. W. Mahastama, "Perbandingan Algoritma *Sum of Squared Difference* (SSD) Dan Optimised *Sum of Absolute Difference* (OSAD) Untuk Pengenalan Simbol Pada Citra Ekspresi Matematika Tercetak," *J. Inform.*, vol. 12, no. 1, 2016, doi: 10.21460/inf.2016.121.457.

[5] A. Mulyanto, A. Apriyadi, and P. Prasetyawan, "Rancang Bangun Game Edukasi 'Matching Aksara Lampung' Berbasis Smartphone Android," *Comput. Eng. Sci. Syst. J.*, vol. 3, no. 1, p. 36, 2018, doi: 10.24114/cess.v3i1.8225.

[6] I. G. N. Suryantara, "Implementasi Deteksi Tepi Untuk Mendeteksi Keretakan Tulang Orang Lanjut Usia (Manula) Pada Citra Rontgen Dengan Operator Sobel Dan Prewitt," *J. Algoritm. Log. dan Komputasi*, vol. 1, no. 2, pp. 51–60, 2018, doi: 10.30813/j-alu.v1i2.1368.

[7] M. S. Ummah, "Dasar Pengolahan Citra Digital Edisi 2022," *Sustain.*, vol. 11, no. 1, pp. 1–14, 2019, [Online]. Available: http://scioteca.caf.com/bitstream/handle/123456789/1091/RED2017-Eng-8ene.pdf?sequence=12&isAllowed=y%0Ahttp://dx.doi.org/10.1016/j.regsciurbeco.2008.06.005%0Ahttps://www.researchgate.net/publication/3053204

84_SISTEM_PEMBETUNGAN_TERPUSAT_STRATEGI_MELESTARI

[8] N. Nafi'iyah, "Algoritma Kohonen dalam Mengubah Citra Graylevel Menjadi Citra Biner," *J. Ilm. Teknol. Inf. Asia*, vol. 9, no. 2, pp. 49–55, 2015, [Online]. Available: https://jurnal.stmikasia.ac.id/index.php/jitika/article/view/125

[9] J. F. Fauzi, H. Tolle, and R. K. Dewi, "Implementasi Metode RGB To HSV pada Aplikasi Pengenalan Mata Uang Kertas Berbasis Android untuk Tuna Netra," vol. 2, no. 6, pp. 2319–2325, 2018.

[10] S. R. Sulistiyanti, F. A. Setyawan, and M. Komarudin, "Pengolahan Citra," *J. Penelit. Pendidik. Guru Sekol. Dasar*, vol. 6, no. August, p. 128, 2016.

[11] C. S. Panchal and A. B. Upadhyay, "Depth Estimation Analysis Using *Sum of Absolute Difference* Algorithm," *Int. J. Adv. Res. Electr.*, vol. 3, no. 1, pp. 6761–6767, 2014, [Online]. Available: www.ijareeie.com

[12] M. M. A. Rosa, G. Paim, L. M. G. Rocha, E. A. C. Costa, and S. Bampi, "Exploring Efficient Adder Compressors for Power-Efficient *Sum of Squared Difference*s Design," no. C, pp. 20–23, 2020.

[13] M. B. Hisham, S. N. Yaakob, R. A. A. Raof, A. B. A. Nazren, and N. M. Wafi, "Template Matching Using *Sum of Squared Difference* and Normalized Cross Correlation," no. December, 2015, doi: 10.1109/SCORED.2015.7449303.

[14] P. Doktor, D. T. Elektro, and F. T. Elektro, "PERINGKASAN VIDEO MENGGUNAKAN DETEKSI SCENE," 2018.

[15] J. Yousefi, "Image Binarization using Otsu Thresholding Algorithm," no. May, 2015, doi: 10.13140/RG.2.1.4758.9284.

[16] M. R. Kusdinar, "SEGMENTASI CITRA DIGITAL KOROSI PIPA PROGRAM STUDI INFORMATIKA," 2018.

[17] M. Santoso, T. Indriyani, and R. E. Putra, "Deteksi Microaneurysms Pada

Citra Retina Mata Menggunakan Matched Filter," *INTEGER J. Inf. Technol.*, vol. 2, no. 2, pp. 59–68, 2017, doi: 10.31284/j.integer.2017.v2i2.180.

[18] L. M. Wisudawati and M. Subali, "Tracking Moving Object Menggunakan Background Subtraction dan Template Matching," *J. Ilm. Komputasi*, vol. 22, no. 1, 2023, doi: 10.32409/jikstik.22.1.3337.

[19] N. N. Shah, K. R. Agarwal, and H. M. Singapuri, "Implementation of *Sum of Absolute Difference* Using Optimized Partial Summation Term Reduction," pp. 192–196, 2013.

[20] J. Kilat, "KLASIFIKASI SENTIMENT ANALYSIS PADA KOMENTAR PESERTA DIKLAT MENGGUNAKAN METODE K-NEAREST NEIGHBOR," vol. 8, no. 1, pp. 81–92, 2019.

[21] R. Khemiri, H. Kibeya, F. E. Sayadi, N. Bahri, M. Atri, and N. Masmoudi, "Optimisation of HEVC motion estimation exploiting SAD and SSD GPU-based implementation," *IET Image Process.*, vol. 12, no. 2, pp. 243–253, 2018, doi: 10.1049/iet-ipr.2017.0474.

[22] W. Widystuti and J. B. B. Darmawan, "Pengaruh jumlah data set terhadap akurasi pengenalan dalam *Deep* convolutional network," *Konf. Nas. Sist. Inf.*, pp. 8–9, 2018.