PENGELOMPOKAN AKSARA JAWA DALAM CITRA DIGITAL DENGAN METODE AGGLOMERATIVE HIERARCHICAL CLUSTERING

SKRIPSI

Oleh:
SENATOR MARCIELIO CHEVIRAY DIANO
NIM. 210605110053

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

PENGELOMPOKAN AKSARA JAWA DALAM CITRA DIGITAL DENGAN METODE AGGLOMERATIVE HIERARCHICAL CLUSTERING

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
SENATOR MARCIELIO CHEVIRAY DIANO
NIM. 210605110053

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2025

HALAMAN PERSETUJUAN

PENGELOMPOKAN AKSARA JAWA DALAM CITRA DIGITAL DENGAN METODE AGGLOMERATIVE HIERARCHICAL CLUSTERING

SKRIPSI

Oleh :
SENATOR MARCIELIO CHEVIRAY DIANO
NIM. 210605110053

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 09 Desember 2025

Pembimbing I,	Pembimbing II,
A'la Syauqi, M. Kom	Nurizal Dwi Priandani, M. Kom
NIP. 19771201 200801 1 007	NIP. 199208302022031001

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

HALAMAN PENGESAHAN

PENGELOMPOKAN AKSARA JAWA DALAM CITRA DIGITAL DENGAN METODE AGGLOMERATIVE HIERARCHICAL CLUSTERING

SKRIPSI

Oleh :
SENATOR MARCIELIO CHEVIRAY DIANO
NIM. 210605110053

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 22 Desember 2025

Susunan Dewan Penguji

Ketua Penguji	: Roro Inda Melani, M. T, M. Sc NIP. 19780925 200501 2 008	( )
Anggota Penguji I	: Khadijah F.H. Holle, M. Kom NIP. 19900626 202203 2 002	( )
Anggota Penguji II	: A'la Syauqi, M. Kom NIP. 19771201 200801 1 007	( )
Anggota Penguji III	: Nurizal Dwi Priandani, M. Kom NIP. 199208302022031001	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Senator Marcielio Cheviray Diano
NIM	: 210605110053
Fakultas / Jurusan	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Pengelompokan Aksara Jawa Dalam Citra Digital Dengan Metode Agglomerative Hierarchical Clustering.

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 22 Desember 2025
Yang membuat pernyataan,

Senator Marcielio Cheviray Diano
NIM.210605110053

MOTTO

... Di balik awan kelabu, matahari tetap bersinar ...

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur ke hadirat Allah SWT, penulis mempersembahkan karya ini kepada:

Keluarga
Yang senantiasa memberikan doa, kasih sayang, serta dukungan selama penulis menempuh Pendidikan.

Dosen Pembimbing dan Dosen Penguji
Yang telah memberikan bimbingan, arahan, serta ilmu pengetahuan dengan penuh kesabaran selama proses penyusunan skripsi ini.

Teman-teman
Yang selalu memberikan semangat, dukungan, dan kebersamaan dalam setiap perjalanan studi.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Pengelompokan Aksara Jawa Dalam Citra Digital Dengan Metode Agglomerative Hierarchical Clustering." dengan baik.

Penyusunan skripsi ini tidak terlepas dari bimbingan, dukungan, serta doa dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis menyampaikan ucapan terima kasih kepada:

1. Prof. Dr. Hj. Ilfi Nur Diana, M.Si., CAHRM., CRMP. selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Agus Mulyono, M.Kes. selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Supriyono, M.Kom. selaku Kepala Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. A'la Syauqi, M.Kom. dan Nurizal Dwi Priandani, M.Kom selaku Dosen Pembimbing I dan II yang telah memberikan bimbingan, arahan, serta motivasi selama proses penyusunan skripsi ini.
5. Roro Inda Melani, M. T, M. Sc. dan Khadijah Fahmi Hayati Holle, M.Kom. selaku Dosen Penguji I dan II atas masukan dan saran yang membangun demi kesempurnaan skripsi ini.
6. Syahiduz Zaman, M.Kom. selaku Dosen Wali yang telah memberikan arahan dan bimbingan selama masa studi.

7. Segenap dosen dan staf akademik Program Studi Teknik Informatika Fakultas Sains dan Teknologi atas ilmu dan pelayanan yang diberikan selama masa studi.
8. Keluarga tercinta atas doa, dukungan, dan motivasi yang tiada henti kepada penulis.
9. Rekan-rekan mahasiswa Teknik Informatika, khususnya angkatan ASTER 21, yang telah menjadi bagian dari perjalanan akademik dan senantiasa memberikan dukungan.
10. Seluruh pihak yang tidak dapat penulis sebutkan satu per satu yang telah membantu dan mendukung penyusunan skripsi ini.

Penulis menyadari bahwa skripsi ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa mendatang. Besar harapan penulis agar skripsi ini dapat memberikan manfaat serta menambah wawasan bagi pembaca.

Malang, 17 Desember 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Diano, Senator Marcielio Cheviray. 2025. **Pengelompokan Aksara Jawa Dalam Citra Digital Dengan Metode *Agglomerative Hierarchical Clustering*.** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) A'la Syauqi, M. Kom (II) Nurizal Dwi Priandani, M. Kom.

Kata kunci: aksara Jawa, pengelompokan, *Agglomerative Hierarchical Clustering*, *Histogram of Oriented Gradients*.

Penelitian ini membahas permasalahan kualitas hasil pengelompokan (*clustering*) aksara Jawa pada citra digital yang memiliki variasi bentuk tinggi serta kualitas visual yang tidak seragam akibat faktor usia dokumen historis. Permasalahan utama dalam penelitian ini adalah bagaimana kualitas hasil *clustering* aksara Jawa menggunakan metode *Agglomerative Hierarchical Clustering* (AHC) dengan ekstraksi fitur *Histogram of Oriented Gradients* (HOG), yang dievaluasi menggunakan metrik *Silhouette Score* dan *Davies Bouldin Index* (DBI). Sebagai solusi, penelitian ini mengusulkan penerapan tahapan *preprocessing* citra yang meliputi *Gaussian Blur*, *Bilateral Filter*, dan *Adaptive Thresholding* untuk meningkatkan kualitas visual citra sebelum dilakukan segmentasi. Data penelitian berupa citra aksara Jawa hasil digitalisasi Majalah Kejawen volume 6 dan 21 tahun 1927. Setelah proses segmentasi, setiap citra aksara tunggal diekstraksi cirinya menggunakan HOG dan selanjutnya dikelompokkan menggunakan metode AHC dengan variasi nilai *distance threshold* sebagai skenario uji coba. Evaluasi kualitas *clustering* dilakukan menggunakan *Silhouette Score* dan Davies–Bouldin Index untuk menilai tingkat pemisahan antar cluster dan kekompakan internal cluster. Hasil pengujian menunjukkan bahwa *distance threshold* 12 memberikan konfigurasi terbaik dengan nilai *Silhouette Score* sebesar 0,0837 dan *Davies Bouldin Index* sebesar 2,7621, serta menghasilkan 159 cluster yang relatif stabil dan representatif terhadap variasi bentuk aksara Jawa. Meskipun nilai evaluasi secara keseluruhan masih tergolong rendah akibat pengaruh kualitas citra historis seperti aksara terputus dan ketidakseragaman tinta, penelitian ini menyimpulkan bahwa metode *Agglomerative Hierarchical Clustering* dengan fitur HOG mampu melakukan pengelompokan aksara Jawa secara cukup baik.

ABSTRACT

Diano, Senator Marcielio Cheviray. 2025. **Clustering of Javanese Script in Digital Images Using the Agglomerative Hierarchical Clustering Method**. Undergraduate Thesis. Department of Informatics Engineering, Faculty of Science and Technology, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Supervisors: (I) A'la Syauqi, M.Kom. (II) Nurizal Dwi Priandani, M.Kom.

Keywords: Javanese script, clustering, Agglomerative Hierarchical Clustering, Histogram of Oriented Gradients.

This study addresses the problem of clustering quality in Javanese script digital images, which exhibit high shape variability and inconsistent visual quality due to the age-related degradation of historical documents. The main problem investigated in this research is how well Javanese script images can be clustered using the Agglomerative Hierarchical Clustering (AHC) method with Histogram of Oriented Gradients (HOG) feature extraction, evaluated using the Silhouette Score and Davies Bouldin Index (DBI) metrics. As a solution, this study proposes the application of image preprocessing stages, including Gaussian Blur, Bilateral Filter, and Adaptive Thresholding, to enhance visual image quality prior to segmentation. The dataset consists of digital images of Javanese script obtained from the digitization of Majalah Kejawen volumes 6 and 21 published in 1927. After segmentation, each single -character image is processed using HOG feature extraction and subsequently clustered using the AHC method with various distance threshold values as experimental scenarios. Clustering performance is evaluated using the Silhouette Score and Davies Bouldin Index to assess inter-cluster separation and intra-cluster compactness. Experimental results show that a distance threshold of 12 yields the best configuration, achieving a Silhouette Score of 0.0837 and a Davies Bouldin Index of 2.7621, resulting in 159 clusters that are relatively stable and representative of the variations in Javanese script shapes. Although the overall evaluation scores remain relatively low due to limitations inherent in historical image quality such as broken characters and inconsistent ink thickness. This study concludes that Agglomerative Hierarchical Clustering combined with HOG features is capable of performing Javanese script clustering with reasonably good performance.

البحث مستخلص

ديانو، سيناتور مارسيليو تشيفيراي. ٢٠٢٥. تجميع الأحرف الجاوية في الصور الرقمية باستخدام طريقة التجميع الهرمي التكتلي (Agglomerative Hierarchical Clustering). بحث جامعي. قسم تقنية المعلومات، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية بمالانج. المشرفان: (١) أعلى شوقي، الماجستير (٢) نوريزال دوي برياندايني، الماجستير

الكلمات المفتاحية :الأحرف الجاوية، التجميع (Clustering)، التجميع الهرمي التكتلي (AHC)، مدرج التدرجات الموجهة (HOG).

يناقش هذا البحث مشكلة جودة نتائج تجميع (Clustering) الأحرف الجاوية في الصور الرقمية التي تتميز بتباين عالٍ في الأشكال وجودة بصرية غير موحدة نتيجة لعامل عمر الوثائق التاريخية .تتمثل المشكلة الرئيسية في هذا البحث في تحديد جودة نتائج تجميع الأحرف الجاوية باستخدام طريقة التجميع الهرمي التكتلي (AHC) مع استخراج الميزات باستخدام مدرج التدرجات الموجهة (HOG)، والتي يتم تقييمها باستخدام مقياس "Davies Bouldin Index" (DBI). ومؤشر "Silhouette Score" وكحل مقترح، يقدم هذا البحث تطبيق مراحل المعالجة المسبقة للصور (Preprocessing) التي تشمل "Gaussian Blur" لتحسين الجودة البصرية للصورة قبل إجراء عملية التقطيع "Adaptive Thresholding"و "Bilateral Filter"و (Segmentation). تتكون بيانات البحث من صور للأحرف الجاوية الناتجة عن رقمنة مجلة "Kejawen" المجلدين ٦ و ٢١ لعام ١٩٢٧ .بعد عملية التقطيع، يتم استخراج خصائص كل صورة حرف منفردة باستخدام تقنية HOG، ومن ثم تجميعها باستخدام طريقة AHC مع تنويع قيم عتبة المسافة (distance threshold) باستخدام كسيناريوهات تجريبية .تم إجراء تقييم جودة التجميع لقياس مستوى الفصل بين المجموعات والتماسك الداخلي للمجموعة "Davies Bouldin Index"و "Silhouette Score". أظهرت نتائج الاختبار أن عتبة المسافة بقيمة ١٢ تعطي أفضل تكوين، حيث بلغت قيمة "Silhouette Score" ٠,٠٨٣٧ مستقرة نسبياً وممثلة لتباين أشكال (cluster) ٢,٧٦٢١، كما أنتجت ١٥٩ مجموعة "Davies Bouldin Index" وقيمة الأحرف الجاوية .على الرغم من أن قيم التقييم الإجمالية لا تزال منخفضة نسبياً بسبب تأثير جودة الصور التاريخية مثل الأحرف قادرة HOG مع ميزات (AHC) المقطوعة وعدم انتظام الحبر، إلا أن هذا البحث يخلص إلى أن طريقة التجميع الهرمي التكتلي على تجميع الأحرف الجاوية بشكل جيد.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Aksara Jawa merupakan sistem tulisan tradisional yang digunakan untuk menuliskan bahasa Jawa (Damayanti, 2019). Aksara ini memiliki nilai historis dan budaya yang tinggi karena banyak naskah kuno, prasasti, serta catatan sejarah Nusantara ditulis menggunakan aksara tersebut. Namun saat ini, penggunaan aksara Jawa mengalami penurunan yang signifikan. Masyarakat kini lebih banyak menggunakan aksara Latin untuk berkomunikasi maupun menulis dokumen (Himamunanto & Widiarti, 2017). Akibatnya, banyak generasi muda yang tidak lagi mengenal, memahami, ataupun membaca aksara Jawa secara mandiri.

Upaya pelestarian aksara Jawa ini sejalan dengan perintah Allah dalam Al-Qur'an agar manusia mengambil pelajaran dari umat terdahulu. Sebagaimana firman-Nya dalam QS. Ar-Rum ayat 9:

اَوَلَمْ يَسِيْرُوْا فِى الْاَرْضِ فَيَنْظُرُوْا كَيْفَ كَانَ عَاقِبَةُ الَّذِيْنَ مِنْ قَبْلِهِمْۗ كَانُوْٓا اَشَدَّ مِنْهُمْ قُوَّةً وَّاَثَارُوا الْاَرْضَ وَعَمَرُوْهَآ

اَكْثَرَ مِمَّا عَمَرُوْهَا وَجَآءَتْهُمْ رُسُلُهُمْ بِالْبَيِّنٰتِۗ فَمَا كَانَ اللّٰهُ لِيَظْلِمَهُمْ وَلٰكِنْ كَانُوْٓا اَنْفُسَهُمْ يَظْلِمُوْنَ ۝٩

"Tidakkah mereka bepergian di bumi lalu memperhatikan bagaimana kesudahan orang-orang sebelum mereka? Mereka itu lebih hebat kekuatannya daripada mereka, dan mereka telah mengolah bumi serta memakmurkannya lebih banyak daripada apa yang telah mereka makmurkan. Dan telah datang kepada mereka rasul-rasul mereka dengan membawa bukti-bukti yang nyata. Maka Allah tidak berlaku zalim kepada mereka, tetapi merekalah yang berlaku zalim kepada diri mereka sendiri." (QS. Ar-Rum: 9)

Ayat ini menegaskan pentingnya mempelajari peninggalan sejarah sebagai bentuk refleksi terhadap perjalanan peradaban manusia. Di Indonesia, salah satu wujud peninggalan tersebut adalah aksara Jawa yang merekam nilai-nilai sosial, budaya, dan pengetahuan lokal. Oleh karena itu, menjaga serta mempelajari aksara Jawa merupakan bagian penting dalam upaya pelestarian warisan intelektual bangsa.

Meskipun pembelajaran aksara Jawa masih diajarkan melalui muatan lokal di berbagai jenjang pendidikan, efektivitasnya dinilai belum optimal dalam meningkatkan pemahaman dan pelestarian aksara Jawa di kalangan masyarakat luas (Fakhruddin et al., 2023). Kondisi ini menunjukkan perlunya pendekatan yang lebih inovatif supaya aksara Jawa lebih mudah dipelajari dan diakses lebih luas oleh generasi modern. Salah satu solusi untuk mengatasi permasalahan tersebut adalah pemanfaatan teknologi digital, khususnya melalui proses digitalisasi aksara Jawa yang dikombinasikan dengan sistem *Optical Character Recognition* (OCR) untuk mengenali dan mengonversi aksara Jawa ke dalam bentuk aksara latin. Namun, penerapan teknologi OCR pada aksara Jawa menghadapi berbagai tantangan. Terutama disebabkan oleh tingginya variasi bentuk huruf, adanya *noise*, kompleksitas struktur goresan, distorsi bentuk, serta kualitas citra naskah yang rendah dan tidak seragam (Khan, Adnan, & Basar, 2018).

Salah satu pendekatan untuk mengatasi tantangan tersebut adalah dengan melakukan pengelompokan citra aksara berdasarkan kesamaan bentuk dan karakteristik visualnya. Proses pengelompokan ini berfungsi untuk mengorganisir huruf-huruf yang memiliki kemiripan sehingga dapat mempermudah proses

pelabelan dan penyusunan dataset pelatihan bagi sistem pengenalan karakter (Peikari et al., 2018). Metode *Agglomerative Hierarchical Clustering* (AHC) merupakan salah satu teknik yang efektif karena mampu mengelompokkan data tanpa harus menentukan jumlah cluster di awal serta memberikan representasi hierarkis kemiripan antar huruf (Oti & Olusola, 2024).

Penelitian ini bertujuan mengelompokan aksara Jawa dalam citra digital dengan metode *Agglomerative Hierarchical Clustering* untuk membentuk dataset yang terstruktur. Hasil dari pengelompokan ini dapat dijadikan dasar untuk pengembangan sistem pengenalan dan transliterasi aksara Jawa secara otomatis.

1.2 Pernyataan Masalah

Berdasarkan latar belakang yang telah diuraikan, pernyataan masalah dalam penelitian ini adalah bagaimana kualitas hasil *clustering* aksara Jawa pada citra digital menggunakan metode *Agglomerative Hierarchical Clustering* (AHC) dengan ekstraksi fitur *Histogram of Oriented Gradients* (HOG) dengan metrik evaluasi menggunakan Silhoutte Score dan *Davies Bouldin Index*?

1.3 Batasan Masalah

Dalam penelitian ini, terdapat beberapa batasan yang diterapkan untuk menjaga fokus dan ruang lingkup yang jelas. Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Jenis Citra yang Digunakan.

Penelitian ini menggunakan citra aksara Jawa yang berasal dari gambar Majalah Kejawen edisi 6 dan 21 yang diterbitkan tahun 1927. Citra ini memiliki

kualitas yang bervariasi, dengan kemungkinan adanya gangguan seperti *noise* atau blur. Gambar yang digunakan adalah gambar dalam format raster (seperti PNG, JPG, atau JPEG), dan *preprocessing* dilakukan untuk memperbaiki kualitas gambar agar dapat diproses lebih lanjut.

2. Jenis Aksara Jawa yang Digunakan.

Penelitian ini akan fokus pada aksara Jawa standar yang digunakan dalam teks-teks yang terdapat pada Majalah Kejawen lama. Aksara yang dimaksud termasuk aksara Jawa utama beserta sandangan seperti pangkon dan tanda baca lainnya yang lazim digunakan dalam penulisan aksara Jawa.

1.4 Tujuan Penelitian

Tujuan penilitian ini adalah mengetahui kualitas hasil *clustering* aksara Jawa pada citra digital menggunakan metode *Agglomerative Hierarchical Clustering* (AHC) dengan ekstraksi fitur *Histogram of Oriented Gradients* (HOG) dengan metrik evaluasi menggunakan Silhoutte Score dan *Davies Bouldin Index*.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat praktis dan teoritis. Secara praktis, penelitian ini dapat mendukung pelestarian aksara Jawa dengan mengembangkan teknologi untuk mempermudah pengolahan dan digitalisasi aksara Jawa dalam citra gambar. Sistem pengelompokan aksara yang dikembangkan dapat digunakan untuk menyusun dan mengelompokkan teks-teks bersejarah, serta mendukung pengembangan sistem translasi aksara Jawa berbasis machine learning.

Secara teoritis, penelitian ini memperkaya kajian pengolahan citra dan kecerdasan buatan, khususnya dalam pengenalan aksara tradisional. Penelitian ini juga membuka peluang untuk pengembangan digitalisasi budaya, yang dapat membantu pelestarian dan akses informasi di masa depan.

BAB II

STUDI PUSTAKA

2.1 Segmentasi

Segmentasi adalah proses dalam pengolahan citra yang bertujuan untuk memisahkan elemen-elemen penting dari gambar, seperti teks, objek, atau fitur lainnya (Tanaya & Adriani, 2018). Dalam konteks pengolahan citra teks, segmentasi merujuk pada pemisahan karakter-karakter yang ada dalam sebuah citra teks menjadi bagian-bagian terpisah yang dapat dianalisis lebih lanjut. Proses segmentasi ini sangat penting dalam aplikasi seperti pengenalan karakter optik (OCR)**,** transliterasi otomatis, dan analisis citra teks (Afakh et al., 2018).

Pada Aksara Jawa, segmentasi menjadi tantangan tersendiri karena aksara ini memiliki bentuk dan struktur yang unik, dengan karakter dasar yang dapat disertai oleh sandangan (penanda vokal) yang diletakkan di berbagai zona di sekitar karakter utama. Oleh karena itu, segmentasi yang tepat dan akurat sangat penting agar karakter aksara Jawa dapat diproses lebih lanjut untuk dikenali dan ditransliterasi ke aksara Latin.

Sebelum membahas lebih lanjut tentang metode segmentasi yang digunakan dalam penelitian ini, perlu untuk memahami bagaimana penelitian terdahulu mengaplikasikan berbagai teknik segmentasi dalam pengolahan teks. Penelitian-penelitian tersebut mencakup berbagai metode yang digunakan untuk mendeteksi karakter teks, terutama dalam pengolahan Aksara Jawa dan jenis teks lainnya. Setiap metode memiliki karakteristik dan hasil yang berbeda-beda, tergantung pada tipe citra yang digunakan dan kondisi data. Oleh karena itu, pemahaman terhadap

hasil-hasil penelitian tersebut akan memberikan gambaran yang lebih jelas tentang bagaimana teknik segmentasi bekerja dalam berbagai kondisi dan pengaruhnya terhadap akurasi deteksi karakter.

Tabel 2. 1 Penelitian Segmentasi Terdahulu

No.	Judul Penelitian	Penulis	Metode	Hasil	Tantangan
1.	Word Segmentation for Javanese Character using Dictionary, SVM, and CRF	Dipta Tanaya, Mirna Adriani	SVM, CRF, Dictionary	Kombinasi metode menghasilkan F1-measure 0.904 dengan akurasi tinggi pada segmentation.	Ambiguity pada afiks dan entitas bernama.
2.	Javanese Character Image Segmentation of Document Image of Hamong Tani	Agustinus Rudatyo Himamunanto, Anastasia Rita Widiarti	Proyeksi Profil	Akurasi segmentasi rata-rata 84.255% pada buku Hamong Tani.	Kesalahan segmentasi karena kerusakan gambar atau karakter yang saling menempel.
3.	Aksara Jawa Text Detection in Scene Images using Convolutional Neural Network	Muhammad Labiyb Afakh, Anhar Risnumawan, Martianda Erste Anggraeni	CNN	Segmentasi karakter lebih baik dengan CNN dalam gambar dengan latar belakang kompleks.	Kualitas gambar rendah mempengaruhi akurasi.
4.	Segmentation of Javanese Characters in Ancient Manuscript using Connected Component Labeling	Fitri Damayanti, Yoyon Kusnendar Suprapto, Eko Mulyanto Yuniarno	Connected Component Labeling (CCL)	Akurasi segmentasi mencapai 80.9% dengan metode CCL.	Karakter yang saling menempel atau buram menyebabkan kesalahan segmentasi.

Berdasarkan beberapa penelitian sebelumnya mengenai segmentasi Aksara Jawa, terdapat empat penelitian yang dapat dibandingkan sebagaimana ditunjukkan pada Tabel 2.1, dengan penjelasan sebagai berikut:

1. Tanaya dan Adriani (2018) menggunakan kombinasi metode *Support Vector Machine* (SVM), *Conditional Random Field* (CRF), dan *Dictionary*-

based segmentation untuk melakukan segmentasi kata pada teks Aksara Jawa. Metode ini memanfaatkan kemampuan SVM dalam mengklasifikasikan fitur tekstual dan CRF dalam mempertahankan hubungan kontekstual antar karakter. Hasil penelitian menunjukkan nilai F1-measure sebesar 0.904, menandakan akurasi yang tinggi dalam segmentasi kata Aksara Jawa. Namun, penelitian ini masih menghadapi kendala pada penanganan afiks dan struktur morfologi yang kompleks.

2. Damayanti, Kusnendar Suprapto, dan Yuniarno (2019) menggunakan metode *Connected Component Labeling* (CCL) untuk melakukan segmentasi karakter pada manuskrip kuno Aksara Jawa. Metode ini bekerja dengan mengelompokkan piksel-piksel yang saling terhubung dalam citra biner menjadi satu komponen terpisah sehingga tiap karakter dapat diidentifikasi secara individual. Hasil penelitian menunjukkan metode ini cukup efektif untuk karakter yang terpisah dengan jelas, namun masih kurang optimal ketika karakter saling menempel.
3. Himamunanto dan Widiarti (2017) menggunakan metode *Projection Profile* untuk melakukan segmentasi karakter pada naskah Aksara Jawa. Proses ini dilakukan dengan menghitung distribusi intensitas piksel secara horizontal dan vertikal untuk menemukan batas antar karakter. Hasil penelitian menunjukkan metode ini cukup akurat pada citra dengan pemisahan karakter yang jelas, namun kurang efektif untuk citra dengan tingkat degradasi tinggi.

4. Sulistyaningsih dan Hidayatno (2021) menerapkan pendekatan *Convolutional Neural Network* (CNN) dalam proses segmentasi karakter Aksara Jawa. Dengan memanfaatkan kemampuan CNN dalam mengenali pola spasial dan fitur visual yang kompleks, penelitian ini mampu meningkatkan akurasi segmentasi, terutama pada citra dengan variasi bentuk dan ukuran aksara yang beragam.

Setiap metode segmentasi tersebut memiliki kelebihan dan kekurangannya masing-masing, tergantung pada kondisi data serta kompleksitas karakter yang digunakan dalam penelitian.

2.2 *Clustering*

Clustering merupakan teknik yang digunakan untuk mengelompokkan data ke dalam beberapa grup atau kluster berdasarkan kesamaan fitur yang dimilikinya (Hussain & Bashir, 2016). Dalam konteks pengolahan citra, teknik ini digunakan untuk mengelompokkan piksel atau objek yang memiliki kesamaan fitur, seperti warna, intensitas, dan tekstur. *Clustering* sangat berguna dalam proses segmentasi citra, karena memungkinkan pemisahan objek-objek dalam gambar berdasarkan kemiripan fitur-fitur yang ada (Wang et al., 2024). Pada Aksara Jawa, *clustering* memiliki tantangan tersendiri karena karakter-karakter dalam aksara ini sering kali saling terkait erat dan memiliki struktur yang sangat unik. Selain itu, aksara ini memiliki sandangan (penanda vokal) yang dapat diletakkan di berbagai posisi di sekitar karakter utama, membuat proses pemisahan antar karakter menjadi lebih sulit. Oleh karena itu, pemilihan teknik *clustering* yang tepat sangat penting untuk

memperoleh hasil segmentasi yang akurat dan efisien, khususnya pada citra yang kualitasnya bervariasi.

Berbagai teknik *clustering* telah diterapkan dalam pengolahan citra teks, termasuk metode *K-Means*, *Agglomerative Hierarchical Clustering* (AHC), dan *Shared Nearest Neighbors* (SNN). Masing-masing metode ini memiliki keunggulan dan kelemahan tergantung pada jenis data citra dan tujuan segmentasi yang diinginkan.

Tabel 2. 2Penelitian *Clustering* Terdahulu

No.	Judul Penelitian	Penulis	Metode	Hasil	Tantangan
1.	Image *Clustering* Based on a Shared Nearest Neighbors Approach for Tagged Collections	Pierre-Alain Moëllic, Jean-Emmanuel Haugeard, Guillaume Pittel	Shared Nearest Neighbors (SNN)	SNN memberikan hasil *clustering* yang lebih representatif untuk koleksi gambar yang besar dan heterogen.	Ketergantungan pada pemilihan parameter (Focus dan Coverage) untuk hasil yang optimal dan menghindari over-*clustering*.
2.	Efficient Enhanced K-Means *Clustering* Algorithm	Ahmed M. Fahim, Abdel-Badeeh M. Salem, Mohamed A. Ramadan	K-Means Enhanced	Peningkatan efisiensi komputasi K-Means memungkinkan *clustering* pada dataset besar dengan hasil yang baik.	Ketergantungan pada pemilihan centroid awal dan proses perhitungan jarak yang memakan waktu.
3.	K-Means Cluster Analysis for Image Segmentation	S. M. Aqil Burney, Humera Tariq	K-Means	K-Means memberikan segmentasi citra yang baik dengan akurasi tinggi pada ruang warna Lab dibandingkan dengan RGB.	K-Means rentan terhadap masalah minim lokal, *noise*, dan over-segmentasi pada citra yang lebih kompleks.

No.	Judul Penelitian	Penulis	Metode	Hasil	Tantangan
4.	Graphical Image Region Extraction with K-Means *Clustering* and Watershed	Sandra Jardim, João António, Carlos Mora	K-Means, Watershed	Kombinasi K-Means dan Watershed efektif untuk ekstraksi region pada gambar dengan variabilitas tinggi.	Tantangan dalam menangani gambar dengan *noise* atau latar belakang yang tidak terstruktur.
5.	An Efficient and Effective Generic *Agglomerative Hierarchical Clustering* Approach	Julien Ah-Pine	*Agglomerative Hierarchical Clustering*	Mengusulkan varian AHC generik yang lebih efisien dan efektif, dengan peningkatan performa dalam akurasi dan waktu komputasi pada dataset besar.	Tantangan dalam menjaga efisiensi ketika jumlah data sangat besar serta pemilihan metode *linkage* yang tepat untuk setiap jenis data.

1. Moëllic et al. mengusulkan pendekatan *Shared Nearest Neighbors* (SNN) untuk pengelompokan koleksi gambar berskala besar. Hasil penelitian tersebut menunjukkan bahwa metode SNN mampu memberikan hasil *clustering* yang lebih representatif pada data yang heterogen, namun masih bergantung pada pemilihan parameter (focus dan coverage) agar tidak terjadi over-*clustering*.
2. Fahim et al. melakukan pengembangan terhadap algoritma *K-Means* melalui peningkatan efisiensi komputasi dengan cara mengoptimalkan proses inisialisasi centroid. Metode ini terbukti mempercepat waktu komputasi dan tetap memberikan hasil yang akurat untuk dataset besar, meskipun masih bergantung pada pemilihan titik awal centroid dan sensitif terhadap outlier.

3. Burney dan Tariq menerapkan algoritma *K-Means* untuk segmentasi citra pada ruang warna Lab, dan hasilnya menunjukkan peningkatan akurasi dibandingkan dengan ruang warna RGB. Akan tetapi, metode ini masih menghadapi kendala pada kasus citra kompleks yang mengandung *noise* dan rentan terhadap local minima.
4. Jardim et al. mengombinasikan metode *K-Means* dan *Watershed* untuk melakukan ekstraksi region pada citra dengan variasi tinggi. Kombinasi kedua metode ini mampu menghasilkan segmentasi yang baik pada citra dengan kontur kompleks, tetapi masih menghadapi tantangan ketika menangani citra dengan background tidak terstruktur atau *noise* yang tinggi.
5. Ah-Pine memperkenalkan pendekatan *Agglomerative Hierarchical Clustering* (AHC) yang generik, efisien, dan efektif untuk berbagai jenis data. Dalam penelitiannya, AHC menunjukkan kemampuan untuk menghasilkan pengelompokan yang stabil tanpa perlu menentukan jumlah cluster di awal, serta fleksibel terhadap bentuk dan ukuran data yang berbeda. Namun demikian, AHC masih memiliki tantangan dalam hal efisiensi ketika jumlah data meningkat secara signifikan, terutama dalam pemilihan jenis *linkage* yang paling sesuai dengan karakteristik dataset.

Secara keseluruhan, pemilihan metode *clustering* yang tepat sangat bergantung pada karakteristik dan kompleksitas data yang digunakan. Metode *Shared Nearest Neighbors* (SNN) lebih sesuai untuk pengelompokan citra dengan kluster yang memiliki ukuran dan bentuk bervariasi, sementara *K-Means* lebih efektif untuk data dengan segmentasi yang sederhana dan struktur kluster yang

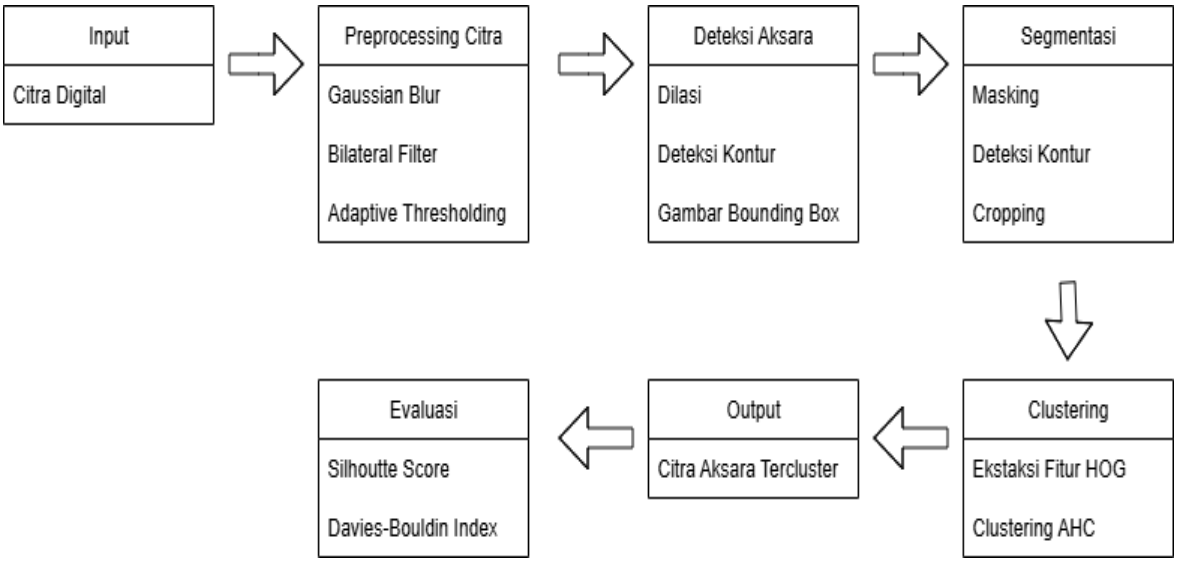
jelas. Di sisi lain, *Agglomerative Hierarchical Clustering* (AHC) memberikan fleksibilitas lebih tinggi dalam mengelompokkan data yang kompleks dan bersifat hierarkis. Berdasarkan hasil kajian literatur, AHC menunjukkan kemampuan untuk membentuk pengelompokan yang stabil tanpa perlu menentukan jumlah kluster di awal serta mampu menyesuaikan diri terhadap bentuk dan ukuran data yang beragam. Oleh karena itu, memahami kelebihan dan keterbatasan masing-masing metode memberikan wawasan penting dalam menentukan teknik *clustering* yang optimal untuk penelitian ini, khususnya dalam pengelompokan aksara Jawa pada citra digital yang memiliki variasi bentuk aksara dan kualitas citra yang tidak seragam.

BAB III

METODE PENELITIAN

3.1 Desain Sistem

Desain sistem yang dikembangkan dalam penelitian ini bertujuan untuk mengelompokkan aksara Jawa dalam citra gambar secara otomatis. Sistem ini dirancang untuk mengatasi tantangan dalam mengenali aksara Jawa yang memiliki variasi bentuk, ukuran, dan kualitas citra yang berbeda. Adapun alur kerja sistemnya seperti pada Gambar 3.1.



Gambar 3. 1 Desain Sistem

Dari Gambar 3.1 terlihat bahwa sistem dimulai dengan menerima citra digital yang berisi aksara Jawa. Citra ini bisa berasal dari dokumen fisik yang dipindai atau gambar digital lainnya.

Setelah citra diterima, tahap pertama yang dilakukan adalah *preprocessing*, di mana citra akan diproses untuk meningkatkan kualitasnya. Proses ini meliputi

penghilangan *noise*, peningkatan kontras, dan penyesuaian kualitas gambar, yang bertujuan agar aksara lebih jelas dan mudah dikenali oleh sistem.

Selanjutnya, pada tahap deteksi aksara, sistem akan memindai citra untuk mendeteksi dan memisahkan aksara Jawa dari latar belakang, serta menandai lokasi aksara yang terdeteksi. Setelah aksara terdeteksi, citra akan melalui tahap segmentasi dan *Cropping*, di mana aksara-aksara yang terpisah akan dipotong menjadi bagian-bagian individu, masing-masing mewakili satu aksara yang akan dikelompokkan lebih lanjut.

Tahap berikutnya adalah *clustering*, di mana aksara yang telah dipisahkan akan dikelompokkan menggunakan metode *Agglomerative Hierarchical Clustering* (AHC). Metode ini akan mengelompokkan aksara berdasarkan kemiripan bentuk dan karakteristik visualnya, tanpa memerlukan penentuan jumlah kluster sebelumnya. AHC dipilih karena fleksibilitasnya dalam mengelompokkan data yang memiliki variasi bentuk.

Setelah proses pengelompokan selesai, sistem menghasilkan output berupa kelompok-kelompok aksara yang terorganisir berdasarkan kesamaan bentuk dan karakteristik visual. Hasil *clustering* ini kemudian dievaluasi menggunakan metrik *Silhouette Score* dan *Davies–Bouldin Index* (DBI) untuk menilai kualitas pemisahan dan kepadatan antar cluster. Tahapan evaluasi ini bertujuan memastikan bahwa pengelompokan yang dihasilkan memiliki validitas dan koherensi yang tinggi. Dataset terstruktur yang telah melalui tahap evaluasi tersebut selanjutnya dapat dimanfaatkan sebagai data latih bagi model machine learning.

3.2 Karakteristik Data

Sebelum dilakukan proses pengolahan citra dan pengelompokan aksara, tahap awal penelitian ini dimulai dengan pengumpulan dan penentuan sumber data yang akan digunakan. Data yang digunakan dalam penelitian ini merupakan citra digital aksara Jawa yang diperoleh dari hasil digitalisasi Majalah Kajawen, sebuah publikasi berbahasa Jawa yang terbit pada awal abad ke-20. Sumber data diambil dari arsip daring Wikisource melalui tautan: https://jv.wikisource.org/wiki/Barkas:Kajawen_06_1927-02-10.pdf.

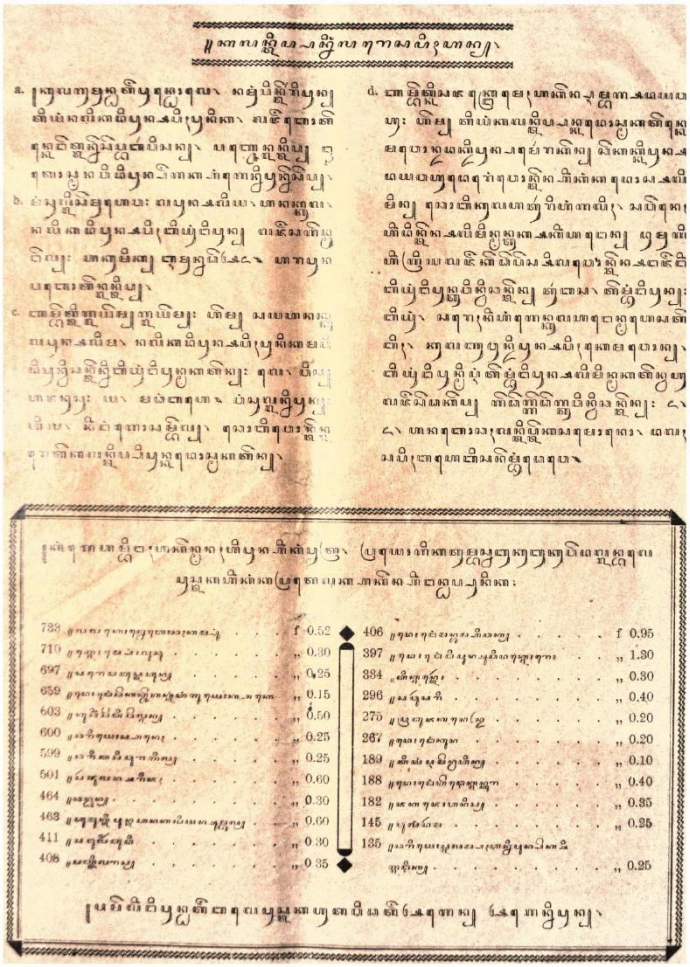
Dataset yang digunakan terdiri atas dua edisi majalah Kajawen, di mana setiap majalah memiliki 20 halaman berisi teks beraksara Jawa dalam berbagai variasi tata letak dan ukuran huruf. Pemilihan sumber ini didasarkan pada keaslian bentuk aksara dan keragaman karakter yang muncul dalam konteks teks nyata, sehingga sesuai untuk keperluan analisis dan pengelompokan aksara secara otomatis menggunakan metode *Agglomerative Hierarchical Clustering* (AHC).

Data diperoleh dalam format PDF hasil pemindaian yang kemudian dikonversi menjadi citra berformat .PNG untuk setiap halaman dengan resolusi rata-rata 150 dpi dan ukuran 1275 × 1650 piksel dengan kedalaman warna 24 bit. Spesifikasi ini dinilai cukup untuk mempertahankan detail visual aksara Jawa tanpa menghasilkan ukuran berkas yang terlalu besar.

Beberapa citra menunjukkan variasi kualitas akibat proses pemindaian dokumen lama, seperti adanya *noise*, variasi pencahayaan, dan ketidakteraturan tepi huruf. Kondisi ini menimbulkan tantangan dalam proses pengolahan citra, terutama pada tahap deteksi dan segmentasi aksara. Oleh karena itu, dilakukan tahap

preprocessing menggunakan *Gaussian Blur* untuk mengurangi *noise*, *Bilateral Filter* untuk mempertahankan tepi aksara, serta *Adaptive Thresholding* untuk meningkatkan kontras antara aksara dan latar belakang.

Contoh data yang digunakan dalam penelitian ini ditunjukkan pada Gambar 3.2, yang memperlihatkan cuplikan halaman Majalah Kajawen hasil digitalisasi yang berisi teks beraksara Jawa.



Gambar 3. 2 Contoh Gambar Majalah Kejawen

Adapun aksara Jawa sendiri memiliki struktur yang kompleks dan bersifat syllabic script. Sistem tulisannya terdiri atas 20 aksara nglegena sebagai huruf

dasar, 12 sandhangan yang berfungsi sebagai penanda vokal atau bunyi tambahan, serta 20 pasangan yang digunakan untuk mematikan konsonan sebelumnya dalam satu suku kata (Sugianela & Suciati, 2019). Kombinasi dari berbagai komponen tersebut menghasilkan banyak variasi bentuk visual yang memiliki kemiripan struktur, namun berbeda fungsi fonetik. Kompleksitas inilah yang menjadikan aksara Jawa menantang untuk dikelompokkan secara otomatis, karena diperlukan metode yang mampu membedakan dan mengelompokkan huruf berdasarkan kemiripan bentuk secara adaptif tanpa bantuan label manual.

3.3 Pengolahan Citra

Pada tahap ini, citra yang berisi aksara Jawa akan diproses melalui serangkaian langkah untuk mempersiapkan data yang diperlukan dalam sistem. Proses pengolahan citra ini sangat penting untuk meningkatkan kualitas gambar dan memastikan aksara dapat terdeteksi dengan baik, meskipun citra memiliki variasi bentuk, ukuran, dan kualitas yang berbeda. Tahapan utama dalam pengolahan citra ini adalah *preprocessing*, deteksi aksara, dan segmentasi.

3.3.1 *Preprocessing* Citra

Proses *preprocessing* bertujuan untuk meningkatkan kualitas citra agar lebih mudah diproses pada tahap berikutnya. Beberapa langkah yang dilakukan dalam *preprocessing* pada penelitian ini adalah sebagai berikut:

1. *Gaussian Blur*

Gaussian Blur digunakan untuk mengurangi *noise* yang terdapat pada citra dengan cara melakukan konvolusi citra terhadap kernel berbasis distribusi

Gaussian (Smolka, Kusnik, & Radlak, 2023). Kernel *Gaussian* dua dimensi didefinisikan sebagai:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2-y^2}{2\sigma^2}\right) \quad (3.1)$$

dengan:

x, y = koordinat relatif piksel terhadap pusat kernel,

σ = standar deviasi yang mengontrol tingkat penyebaran blur.

Proses *Gaussian Blur* dapat dituliskan sebagai operasi konvolusi:

$$I'(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(x-i, y-j).G(i,j) \quad (3.2)$$

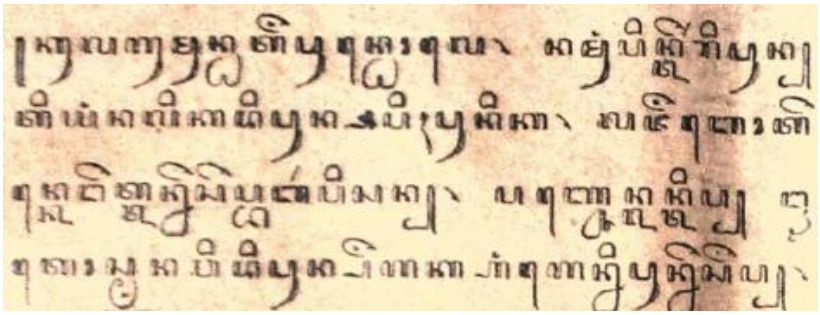
dengan:

$I(x,y)$ = intensitas piksel pada koordinat (x,y),

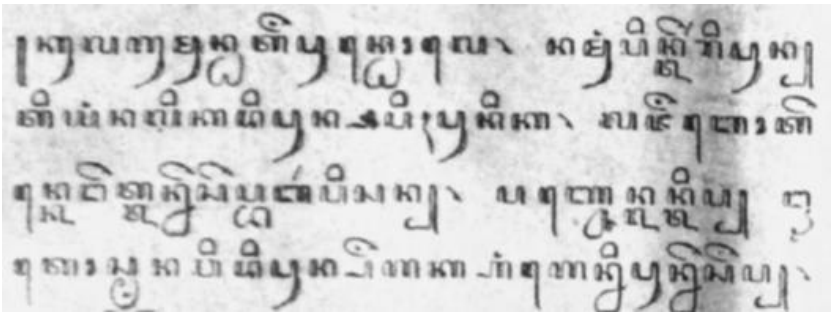
$G(i,j)$ = nilai kernel *Gaussian* pada posisi (i,j),

$I'(x,y)$ = hasil citra terfilter.

Metode ini bekerja dengan menghitung rata-rata berbobot di sekitar setiap piksel menggunakan distribusi *Gaussian* ,sehingga piksel-piksel yang dekat dengan pusat kernel memiliki bobot lebih besar dibandingkan yang jauh. Hasilnya adalah citra yang lebih halus, di mana *noise* berfrekuensi tinggi dapat ditekan, tetapi detail aksara yang lebih besar tetap terjaga. Dengan demikian, penerapan *Gaussian Blur* mempermudah tahap selanjutnya seperti deteksi kontur, segmentasi, maupun *clustering* aksara.



Gambar 3. 3 Input *Preprocessing*



Gambar 3. 4 Output *Gaussian Blur*

2. *Bilateral Filter*

Setelah *Gaussian Blur*, *Bilateral Filter* diterapkan untuk menghaluskan citra sekaligus mempertahankan tepi aksara. Filter ini mempertimbangkan kedekatan spasial dan perbedaan intensitas warna antar piksel (Joseph & Periyasamy, 2018). Rumus matematisnya:

$$I'(x) = \frac{1}{W_p}\sum_{\{x_i \in \Omega\}} I(x_i), f_{r(|I(x_i)- I(x)|)}, f_{s(|x_i - x|)} \quad (3.3)$$

dengan:

I(x): nilai intensitas piksel pada posisi x

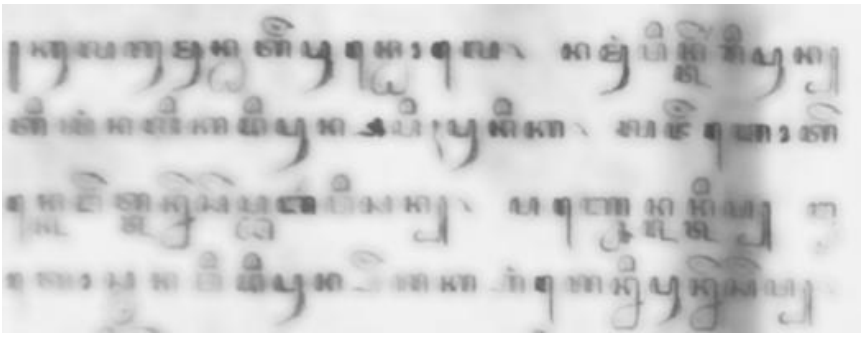
Ω: area jendela di sekitar piksel x

f_r: fungsi *Gaussian* berdasarkan perbedaan intensitas (domain range)

f_s: fungsi *Gaussian* berdasarkan jarak spasial

W_p: faktor normalisasi.

Filter ini menghaluskan area dengan intensitas seragam namun tetap menjaga batas tepi agar tidak ikut mengabur.



Gambar 3. 5 Output *Bilateral Filter*

3. *Adaptive Thresholding*:

Tahap selanjutnya adalah *Adaptive Thresholding*, yang digunakan untuk mengubah citra hasil *filtering* menjadi citra biner. Nilai ambang ($T(x,y)$) dihitung secara adaptif berdasarkan rata-rata lokal di sekitar piksel.

$$T(x,y) = \frac{1}{S^2}\sum_{\{(i,j)\in S\}} I(i,j) - C \tag{3.4}$$

dengan:

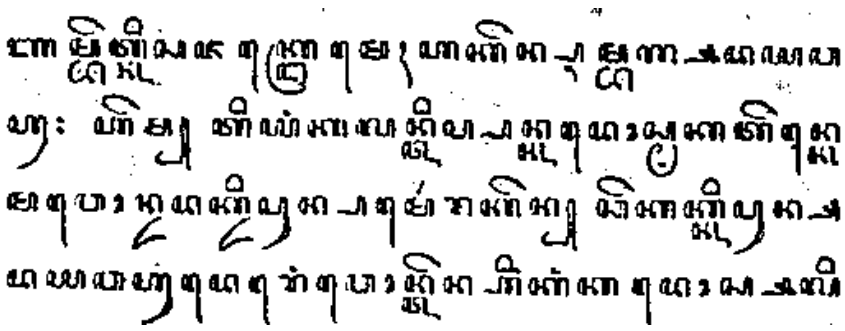
S: ukuran blok atau jendela lokal di sekitar piksel (x, y)

C: konstanta pengurang untuk menyesuaikan sensitivitas threshold.

Kemudian, proses binerisasi dilakukan dengan aturan:

$$I'(x,y) = \begin{cases} 1, jika\ I(x,y) > T\ (x,y) \\ 0,\ lainnya \end{cases} \tag{3.5}$$

Metode ini efektif untuk citra dengan variasi pencahayaan tidak seragam seperti dokumen Majalah Kejawen, karena setiap area citra mendapat nilai ambang berbeda sesuai kondisi lokalnya.



Gambar 3. 6 Output Adaptive Threshold

3.3.2 Deteksi Aksara

Setelah *preprocessing* selesai, tahap berikutnya adalah deteksi aksara. Pada tahap ini, citra yang telah diproses akan diperiksa untuk mendeteksi keberadaan aksara Jawa. Deteksi aksara dilakukan melalui beberapa langkah:

1. Dilasi

Dilasi merupakan operasi morfologi yang berfungsi untuk memperbesar area piksel berwarna terang (foreground) pada citra biner (Said & Jambek, 2021). Dalam konteks deteksi aksara Jawa, dilasi berperan penting untuk menyatukan bagian-bagian aksara yang mungkin terpisah akibat proses segmentasi sebelumnya, seperti pangkon, sandhangan, atau tanda baca lainnya, agar terbentuk satu kesatuan aksara yang utuh.

Secara matematis, operasi dilasi pada citra biner A dengan structuring element B didefinisikan sebagai berikut:

$$A \oplus B = \{ z \mid (\hat{B})_z \cap A \neq \emptyset \} \tag{3.6}$$

Keterangan:

A: himpunan piksel pada citra biner yang merepresentasikan objek (aksara).

B: structuring element (biasanya berbentuk matriks kecil seperti persegi atau lingkaran).

$\hat{B}$: refleksi dari B terhadap pusatnya.

z: vektor translasi yang menunjukkan posisi structuring element.

$\oplus$: operator dilasi.

Secara intuitif, dilasi menambahkan piksel ke tepi objek berdasarkan bentuk structuring element B. Semakin besar ukuran B, semakin besar pula area hasil dilasi. Operasi ini membantu menyatukan bagian-bagian aksara yang terpisah akibat *noise* atau segmentasi yang tidak sempurna, sehingga memperkuat kontur dan struktur aksara sebelum proses deteksi kontur dilakukan.

2. Deteksi Kontur

Setelah proses dilasi, tahap berikutnya adalah deteksi kontur, yaitu proses untuk menemukan batas-batas objek (dalam hal ini aksara Jawa) dalam citra biner. Kontur didefinisikan sebagai kurva yang menghubungkan titik-titik dengan intensitas yang sama (isointensity curve), yang membentuk batas antara objek dan latar belakang (Cai et al., 2024).

Deteksi kontur dilakukan untuk mengidentifikasi dan memisahkan setiap aksara dari citra yang telah melalui tahap *preprocessing*, sehingga sistem dapat mengenali masing-masing aksara secara terpisah.

Secara matematis, kontur dapat direpresentasikan sebagai himpunan titik C pada citra $I\ (x, y)$ yang memiliki nilai intensitas konstan k yaitu:

$$C = \{ (x, y) \mid I(x, y) = k \} \tag{3.7}$$

Pada implementasinya, proses deteksi kontur dilakukan pada citra biner hasil thresholding dengan mencari perbedaan nilai gradien intensitas (∇I).

Gradien intensitas suatu citra dua dimensi didefinisikan sebagai:

$$\nabla I = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \tag{3.8}$$

Sedangkan besarnya gradien diberikan oleh:

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{3.9}$$

Keterangan:

$I(x, y)$: intensitas piksel pada koordinat (x, y).

k: nilai intensitas konstan pada batas objek (isolevel).

∇I: vektor gradien intensitas.

$\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$: turunan parsial terhadap arah horizontal dan vertikal.

$|\nabla I|$: besar gradien, menunjukkan tingkat perubahan intensitas pada suatu titik.

C: kurva atau himpunan titik-titik yang membentuk kontur objek.

3. Gambar *Bounding Box*

Setelah kontur ditemukan, langkah selanjutnya adalah menggambar *bounding box* di sekitar setiap aksara yang terdeteksi. *Bounding box* merupakan persegi yang mengelilingi suatu objek dalam citra berdasarkan batas terluarnya (extreme points) dari hasil deteksi kontur (Kowalczyk et al., 2022). Tahapan ini bertujuan untuk memisahkan setiap aksara secara fisik dari bagian citra lainnya, sehingga setiap aksara, termasuk sandhangan maupun pangkon, dapat dianalisis secara individual pada tahap segmentasi dan *clustering* berikutnya.

Secara matematis, *bounding box* dapat dinyatakan dengan notasi:

$$B_i = \{ (x, y) \mid x_{min}^{(i)} \leq x \leq x_{max}^{(i)}, ; y_{min}^{(i)} \leq y \leq y_{max}^{(i)} \} \quad (3.10)$$

Keterangan:

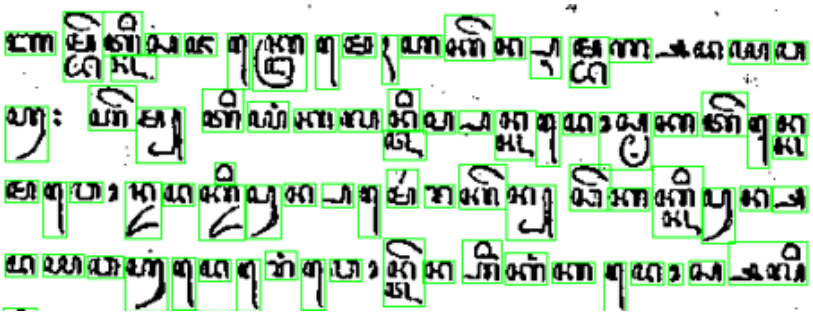
Bi: *bounding box* ke- i yang mengelilingi aksara ke- i.

(x, y): koordinat piksel pada citra.

$x_{min}^{(i)}, x_{max}^{(i)}$: batas minimum dan maksimum koordinat sumbu-X dari kontur aksara ke- i.

$y_{min}^{(i)}, y_{max}^{(i)}$: batas minimum dan maksimum koordinat sumbu-Y dari kontur aksara ke- i.

Dengan demikian, setiap *bounding box* $B\ i$ secara efektif membungkus area citra yang mengandung aksara, memisahkannya dari latar belakang serta objek lain di sekitarnya.



Gambar 3. 7 Output Bounding Box

3.3.3 Segmentasi (*Cropping*)

Proses segmentasi bertujuan untuk memisahkan aksara Jawa menjadi bagian individu yang lebih mudah untuk dikelompokkan pada tahap *clustering*. Setelah tahap deteksi aksara sebelumnya, gambar yang telah diproses dengan *bounding box* akan dipotong berdasarkan langkah-langkah berikut:

1. *Masking* dengan *Bounding box* Hijau:

Pada tahap ini, gambar yang telah diberi *bounding box* hijau akan digunakan untuk memisahkan aksara dari latar belakang. *Masking* dilakukan dengan memanfaatkan informasi dari *bounding box* hijau yang telah digambar sebelumnya. Area yang berada dalam *bounding box* hijau akan dipertahankan, sedangkan area di luar *bounding box* (yang tidak mengandung aksara) akan dihapus. Dengan cara ini, hanya aksara yang terisolasi dalam *bounding box* hijau yang akan diproses pada tahap berikutnya.

Secara matematis, proses *Masking* dapat didefinisikan sebagai:

$$M(x,y) = \begin{cases} I(x,y), jika\ (x,y) \in Bi \\ 0, jika\ (x,y) \notin Bi \end{cases} \quad (3.11)$$

Keterangan:

$M(x,y)$: hasil citra setelah proses *Masking* pada koordinat (x, y).

I (x, y): nilai intensitas piksel dari citra asli pada koordinat (x, y).

$B\ i$: *bounding box* ke- i yang mengelilingi aksara ke- i.

Nilai 0 menunjukkan area yang dihapus.

2. Deteksi Kontur

Setelah *Masking*, langkah berikutnya adalah deteksi kontur. Proses ini bertujuan untuk menemukan batas-batas dari aksara yang telah diisolasi. Kontur digunakan untuk menentukan posisi dan area dari aksara yang terdeteksi. Dengan mendeteksi kontur, kita dapat memisahkan aksara secara lebih akurat, meskipun aksara tersebut saling berdekatan atau memiliki bentuk yang kompleks.

3. *Cropping*

Tahap *Cropping* merupakan proses pemotongan citra berdasarkan koordinat *bounding box* yang telah dihasilkan pada tahap deteksi sebelumnya. Tujuan utama

dari proses ini adalah memisahkan setiap aksara yang terdeteksi menjadi citra individual agar dapat diproses lebih lanjut pada tahap feature extraction atau *clustering*.

Secara matematis, operasi *Cropping* dapat didefinisikan sebagai pemetaan subset dari citra asli $I\ (x,\ y)$ ke dalam citra baru $C\ i\ (x,\ y)$ berdasarkan batas koordinat *bounding box* $B\ i$ untuk aksara ke- i:

$$C_i(x,y) = I(x,y),\ untuk\ x_{min}^{(i)} \leq x\ \leq x_{max}^{(i)}, y_{min}^{(i)} \leq y\ \leq y_{max}^{(i)} \quad (3.12)$$

Keterangan:

$I(x,y)$: citra asli hasil deteksi aksara.

$C_i(x,y)$: citra hasil *Cropping* untuk aksara ke- i.

$x_{min}^{(i)}, x_{max}^{(i)}$: batas minimum dan maksimum koordinat horizontal (sumbu x) dari *bounding box* ke- i.

$y_{min}^{(i)}, y_{max}^{(i)}$: batas minimum dan maksimum koordinat vertikal (sumbu y) dari *bounding box* ke- i.

Dengan demikian, setiap *bounding box* $B\ i$ menghasilkan satu citra terpotong $C\ i$ yang berisi satu aksara Jawa secara utuh. Proses *Cropping* ini penting untuk memastikan bahwa analisis lanjutan, seperti ekstraksi fitur dan pengelompokan menggunakan metode *Agglomerative Hierarchical Clustering*, dapat dilakukan pada objek aksara yang terisolasi tanpa gangguan dari latar belakang atau aksara lain di sekitarnya.

Gambar 3. 8 Hasil *Cropping*

3.4 *Clustering*

Pada tahap *clustering*, tujuan utama adalah mengelompokkan aksara-aksara Jawa berdasarkan kemiripan bentuk dan karakteristik visual yang telah diekstraksi pada tahap sebelumnya. Setelah proses *preprocessing* citra dan deteksi aksara, aksara-aksara yang terdeteksi dalam gambar dipisahkan melalui teknik segmentasi dan diubah menjadi format yang siap untuk dilakukan pengelompokan. Hasil segmentasi ini menghasilkan gambar-gambar individu yang masing-masing mewakili satu aksara Jawa yang dapat dianalisis lebih lanjut.

Pada penelitian ini, metode yang digunakan untuk mengelompokkan aksara Jawa adalah *Agglomerative Hierarchical Clustering* (AHC). AHC adalah teknik *clustering* berbasis hirarki yang mengelompokkan data dengan cara menggabungkan dua kluster yang paling mirip pada setiap iterasi (Ah-Pine, 2018). Keunggulan metode ini adalah tidak memerlukan penentuan jumlah kluster di awal, karena jumlah kluster akan ditentukan berdasarkan jarak antar data.

3.4.1 Proses *Clustering*

Proses *clustering* dilakukan setelah gambar-gambar aksara Jawa terpisah dan siap dianalisis. Fitur-fitur yang digunakan untuk mengelompokkan aksara

adalah fitur *Histogram of Oriented Gradients* (HOG) yang diekstraksi dari citra aksara yang telah diproses pada tahap sebelumnya. HOG merupakan fitur yang sangat baik untuk mengenali bentuk dan tekstur pada gambar, yang memungkinkan identifikasi aksara berdasarkan bentuk visualnya.

1. Ekstraksi Fitur HOG

Tahap ekstraksi fitur menggunakan *Histogram of Oriented Gradients* (HOG) bertujuan untuk merepresentasikan bentuk dan struktur tepi dari aksara Jawa secara numerik, sehingga dapat digunakan sebagai input untuk proses *clustering*. Metode HOG bekerja dengan cara menghitung arah (*orientation*) dan besar (*magnitude*) gradien pada setiap piksel dalam citra, kemudian mengelompokkan nilai tersebut ke dalam histogram berdasarkan arah dominan gradien di dalam setiap sel citra (Deore & Pravin, 2019).

Sebagai ilustrasi, Gambar 3.9 berikut memperlihatkan hasil visualisasi HOG pada salah satu citra aksara Jawa. Terlihat bahwa pola gradien yang terbentuk menyoroti kontur tepi huruf, sehingga informasi bentuk aksara tetap terjaga meskipun tanpa memperhatikan warna atau tekstur asli citra.

Gambar 3. 9 Visualisasi HOG

Secara matematis, gradien horizontal dan vertikal pada citra $I(x, y)$ dapat dihitung dengan operator perbedaan berikut:

$$G_{x(x,y)} = I(x + 1, y) - I(x - 1, y) \tag{3.13}$$

$$G_{y(x,y)} = I(x, y + 1) - I(x, y - 1) \tag{3.14}$$

Berdasarkan kedua komponen tersebut, besar gradien (*magnitude*) dan arah gradien (*orientation*) dihitung menggunakan:

$$M(x,y) = \sqrt{\{G_x^2(x,y) + G_y^2(x,y)\}} \tag{3.15}$$

$$\theta(x,y) = tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \tag{3.16}$$

Setiap sel berukuran 8 × 8 piksel akan menghasilkan histogram dari 9 orientasi gradien $(0^\circ - 180^\circ)$, yang menunjukkan distribusi arah tepi dalam area tersebut. Untuk meningkatkan ketahanan terhadap perubahan pencahayaan, histogram dari beberapa sel yang berdekatan digabungkan ke dalam blok berukuran 2 × 2, dan setiap blok kemudian dinormalisasi dengan norma L2 sebagai berikut:

$$v' = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}} \tag{3.17}$$

Keterangan:

$I\ (x, y)$: citra input dalam format abu-abu.

$G\ x$, $G\ y$: komponen gradien arah horizontal dan vertikal.

$M\ (x, y)$: besar gradien di titik (x, y).

$\theta\ (x, y)$: arah gradien pada titik (x, y).

v: vektor fitur HOG yang belum dinormalisasi.

$v\ '$: vektor fitur HOG setelah normalisasi.

ϵ: konstanta kecil untuk mencegah pembagian dengan nol.

Dengan demikian, setiap citra aksara Jawa diubah menjadi vektor numerik berdimensi tinggi yang menggambarkan pola tepi dan struktur bentuk aksara. Vektor-vektor inilah yang kemudian digunakan sebagai input dalam proses

Agglomerative Hierarchical Clustering (AHC) untuk mengelompokkan aksara berdasarkan kesamaan bentuk visualnya.

2. *Clustering* dengan AHC

Tahap ini bertujuan untuk mengelompokkan citra aksara Jawa berdasarkan kemiripan fitur yang diperoleh dari proses ekstraksi HOG. Metode yang digunakan adalah *Agglomerative Hierarchical Clustering* (AHC), yaitu salah satu pendekatan bottom-up dalam hierarchical *clustering*, di mana setiap data awalnya dianggap sebagai satu cluster tunggal, kemudian secara iteratif digabungkan dengan cluster lain yang paling mirip hingga terbentuk hirarki pengelompokan (Ah-Pine, 2018).

Dalam AHC, kesamaan antar data diukur menggunakan jarak Euclidean antara dua vektor fitur $x\ i$ dan $x\ j$ hasil ekstraksi HOG.

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2} \tag{3.18}$$

Nilai jarak $d(x_i, x_j)$ kemudian digunakan untuk membentuk dendrogram, yaitu struktur hierarki yang menunjukkan hubungan penggabungan antar cluster. Pada setiap iterasi, dua cluster dengan jarak terkecil akan digabungkan menjadi satu cluster baru.

Proses penggabungan ini dikendalikan oleh fungsi *linkage* $D\ (A, B)$, yang menentukan jarak antar dua cluster A dan B. Terdapat beberapa metode *linkage* yang umum digunakan, seperti *single linkage*, *complete linkage*, dan *average linkage*. Dalam penelitian ini digunakan pendekatan *average linkage*, yang dirumuskan sebagai:

$$D(A, B) = \frac{1}{|A||B|}\sum_{i \in A}\sum_{j \in B} d(x_i, x_j) \tag{3.19}$$

Keterangan:

x_i, x_j: vektor fitur HOG dari citra aksara ke- i dan ke- j.

n: jumlah dimensi fitur (jumlah elemen dalam vektor HOG).

$d(x_i, x_j)$: jarak Euclidean antara dua data.

A, B: dua cluster yang akan dibandingkan.

$|A|$, $|B|$: jumlah anggota dalam cluster A dan B.

D (A, B): jarak antar dua cluster menurut metode *average linkage*.

Proses penggabungan ini berlanjut hingga seluruh data tergabung dalam satu cluster besar, membentuk struktur hierarki penuh. Dari dendrogram tersebut, batas cluster dapat ditentukan secara otomatis dengan memotong pohon hirarki pada tingkat jarak tertentu. Hasil akhir dari AHC adalah label cluster yang diberikan untuk setiap citra aksara, di mana label tersebut menunjukkan kelompok aksara yang memiliki kemiripan bentuk visual. Dengan demikian, metode AHC mampu mengelompokkan aksara Jawa yang memiliki karakteristik morfologi serupa tanpa memerlukan penentuan jumlah cluster secara eksplisit di awal.

3. Penyimpanan Hasil *Clustering*

Setelah proses *clustering* selesai, hasilnya adalah gambar-gambar aksara yang terkelompok dalam folder masing-masing sesuai dengan cluster label yang dihasilkan. Setiap folder mewakili satu cluster, dan gambar-gambar yang memiliki kemiripan bentuk akan dikelompokkan bersama di dalam folder yang sama. Ini memungkinkan analisis lebih lanjut terhadap aksara-aksara yang tergolong dalam satu kelompok berdasarkan karakteristik visualnya.

3.5 Evaluasi

Tahap evaluasi dilakukan untuk menilai kualitas hasil *clustering* aksara Jawa yang diperoleh dari metode *Agglomerative Hierarchical Clustering* (AHC). Karena penelitian ini tidak menggunakan data berlabel atau ground truth, maka evaluasi dilakukan menggunakan metrik evaluasi *unsupervised* yang menilai struktur internal data hasil pengelompokan. Metrik yang digunakan dalam penelitian ini adalah *Silhouette Score* dan Davies–Bouldin Index (DBI).

3.5.1 *Silhouette Score*

Silhouette Score digunakan untuk mengukur seberapa baik setiap sampel berada dalam cluster-nya sendiri dibandingkan dengan cluster lain (Hartama & Anjelita, 2022). *Silhouette Score* memberikan nilai antara -1 hingga 1, di mana nilai yang lebih tinggi menunjukkan bahwa cluster semakin baik.

Konsepnya didasarkan pada dua nilai utama untuk setiap data i:

$a(i)$: rata-rata jarak antara titik i dengan semua titik lain dalam cluster yang sama.

b (i): jarak rata-rata minimum antara titik i dan titik-titik pada cluster lain yang paling dekat.

$$s(i) = \frac{b(i) - a(i)}{max\ \{\ a(i), b(i)\}} \tag{3.20}$$

Keterangan:

$s(i)$: nilai Silhouette untuk data ke- i.

a (i): jarak rata-rata antara data ke- i dan semua data dalam cluster yang sama (intra-cluster distance).

b (i): jarak rata-rata antara data ke- i dan data pada cluster terdekat (nearest-cluster distance).

Nilai s (i) berada pada rentang [−1, 1], dengan interpretasi sebagai berikut:

s (i) ≈ 1: data terkelompok dengan baik (berada sangat dekat dengan cluster-nya sendiri dan jauh dari cluster lain).

s (i) ≈ 0: data berada di batas antara dua cluster.

s (i) < 0: data mungkin salah tempat (misclassified), lebih dekat ke cluster lain daripada cluster-nya sendiri.

Untuk keseluruhan hasil *clustering*, nilai rata-rata dari semua $s(i)$ digunakan sebagai *Silhouette Score* total:

$$S = \frac{1}{N}\sum_{i=1}^{N} s(i) \tag{3.21}$$

Keterangan:

S: nilai *Silhouette Score* keseluruhan.

N: jumlah total data yang diklaster.

Semakin tinggi nilai S, semakin baik kualitas *clustering* karena menunjukkan bahwa setiap data berada dekat dengan anggota cluster-nya dan jauh dari cluster lain.

3.5.2 Davies-Bouldin Index

Davies-Bouldin Index (DBI) merupakan metrik evaluasi yang digunakan untuk menilai kualitas hasil *clustering* dengan mempertimbangkan rasio antara dispersi intra-cluster dan jarak antar-cluster (Idrus et al., 2022). Semakin kecil nilai DBI, semakin baik hasil *clustering*, karena menunjukkan bahwa setiap cluster memiliki kepadatan internal tinggi dan terpisah jauh satu sama lain.

Secara matematis, DBI didefinisikan sebagai:

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \underset{i \neq j}{max} \left(\frac{S_i+S_j}{M_{ij}}\right) \quad (3.22)$$

Keterangan:

k = jumlah cluster yang terbentuk.

S_i = rata-rata jarak antara setiap titik dalam cluster ke- i terhadap pusat cluster-nya (C i), yang menggambarkan kepadatan intracluster.

M_{ij} = jarak antara pusat cluster C i dan C j, yang merepresentasikan pemisahan antar cluster.

$d(x, y)$= fungsi jarak (misalnya jarak Euclidean).

Nilai DBI yang rendah menunjukkan bahwa cluster memiliki jarak antar-cluster yang besar dan variasi internal yang kecil, sehingga struktur *clustering* dianggap optimal. Sebaliknya, nilai DBI yang tinggi menandakan bahwa cluster saling tumpang tindih atau kurang terpisah dengan baik.

3.6 Skenario Uji Coba

Uji coba dilakukan menggunakan data citra aksara Jawa yang telah melalui tahap segmentasi, sehingga setiap citra merepresentasikan satu karakter aksara. Seluruh citra hasil segmentasi kemudian diekstraksi cirinya menggunakan metode *Histogram of Oriented Gradients* (HOG). Vektor fitur yang dihasilkan selanjutnya digunakan sebagai input pada algoritma *Agglomerative Hierarchical Clustering* (AHC).

Pengujian dilakukan dengan memvariasikan nilai *distance threshold* untuk mengetahui pengaruhnya terhadap kualitas hasil pengelompokan. Pada setiap nilai *distance threshold*, proses *clustering* dijalankan satu kali dan hasilnya dievaluasi menggunakan dua metrik evaluasi *unsupervised*, yaitu *Silhouette Score* dan *Davies Bouldin Index* (DBI). Kedua metrik ini digunakan untuk menilai keseimbangan

antara tingkat pemisahan antar cluster dan kekompakan cluster yang terbentuk. Sebagai ilustrasi skenario uji coba, contoh hasil pengujian pada beberapa nilai *distance threshold* ditunjukkan pada Tabel berikut.

Tabel 3. 1 Skenario Uji

Distance threshold	***Silhouette Score***	***Davies Bouldin Index***
5	0,0564	1,9434
6	0,056	2,4117
7	0,0585	2,6666
8	0,0625	2,7733
9	0,0651	2,8211
10	0,0666	2,904
11	0,0763	2,8411
12	0,0837	2,7621

Berdasarkan hasil pengujian pada berbagai nilai *distance threshold* tersebut, konfigurasi terbaik ditentukan dengan mempertimbangkan nilai *Silhouette Score* yang relatif tinggi dan nilai *Davies Bouldin Index* yang masih berada pada rentang stabil, serta jumlah cluster yang terbentuk masih memiliki makna representatif. Konfigurasi terbaik ini kemudian dianalisis lebih lanjut pada Bab 4.

BAB IV

HASIL DAN PEMBAHASAN

4.1 *Preprocessing* Citra

Tahap *preprocessing* dilakukan untuk meningkatkan kualitas citra sebelum proses deteksi dan pengelompokan aksara. Proses ini bertujuan mengurangi *noise*, mempertegas tepi aksara, dan menyesuaikan pencahayaan agar karakter dapat terdeteksi dengan lebih akurat. Tahapan *preprocessing* yang diterapkan terdiri atas *Gaussian Blur*, *Bilateral Filter*, dan *Adaptive Thresholding*.

4.1.1 *Gaussian Blur*

Tahap pertama adalah *Gaussian Blur*, yang berfungsi untuk mengurangi *noise* berfrekuensi tinggi pada citra hasil pemindaian. Metode ini bekerja dengan mengonvolusi setiap piksel menggunakan kernel *Gaussian* berukuran ganjil (misalnya 3×3, 5×5, atau 7×7). Semakin besar ukuran kernel, semakin kuat efek penghalusan yang dihasilkan, namun detail kecil pada aksara akan semakin berkurang.

Penerapan *Gaussian Blur* pada citra dilakukan dengan kode program sebagai berikut:

Tabel 4. 1 *Gaussian Blur*

```
# Gaussian Blur
kernel_size = max(1, self.Gaussian _slider["slider"].value() | 1)
blurred = cv2.Gaussian Blur(self.image, (kernel_size,
kernel_size), 0)
```

Parameter kernel_size diatur menggunakan *slider Gaussian Blur*, yang memungkinkan pengguna menyesuaikan tingkat penghalusan secara interaktif. *Slider* ini menerima nilai ganjil antara 1 hingga 11, dan setiap perubahan nilainya langsung memperbarui hasil penghalusan pada citra. Tahapan ini penting agar citra lebih stabil terhadap variasi cahaya sebelum diterapkan filter lanjutan.

4.1.2 *Bilateral Filter*

Setelah proses *Gaussian Blur*, dilakukan *Bilateral Filter* untuk mempertahankan tepi huruf sambil tetap mengurangi *noise* pada area latar belakang. Filter ini mempertimbangkan dua faktor untuk menentukan bobot piksel yaitu Jarak spasial antar piksel (seberapa jauh letaknya), dan Perbedaan intensitas warna antar piksel. Dengan demikian, bagian tepi huruf (yang memiliki perubahan intensitas besar) tidak ikut dihaluskan.

Implementasi *Bilateral Filter* pada penelitian ini ditunjukkan pada kode berikut.

Tabel 4. 2 *Bilateral Filter*

```
# Bilateral Filter
d = self.bilateral_diameter_slider["slider"].value()
sigma = self.bilateral_sigma_slider["slider"].value()
filtered = cv2.bilateralFilter(blurred, d=d, sigmaColor=sigma,
sigmaSpace=sigma)
```

Nilai parameter *Bilateral Filter* ditentukan melalui antarmuka *slider* pada aplikasi *preprocessing*. Nilai awal (*default*) parameter tersebut didefinisikan pada saat inisialisasi *slider*, sebagaimana ditunjukkan pada kode berikut.

Tabel 4. 3 Parameter Bilateral

```
self.bilateral_diameter_slider = self.create_slider(1, 25, 15, 2,
"Bilateral Diameter:")
self.bilateral_sigma_slider = self.create_slider(10, 200, 75, 5,
"Bilateral Sigma:")
```

Berdasarkan inisialisasi tersebut, nilai diameter (d) = 15 dan sigma (σ) = 75 digunakan sebagai konfigurasi awal dalam proses *filtering*. Kombinasi nilai ini diperoleh melalui serangkaian percobaan berulang terhadap berbagai citra aksara Jawa, dengan mempertimbangkan keseimbangan antara kejernihan tepi aksara dan pengurangan *noise* latar belakang. Pada konfigurasi ini, struktur goresan aksara tetap terjaga dengan baik, sementara gangguan visual seperti bintik dan variasi intensitas latar belakang dapat ditekan secara signifikan.

4.1.3 *Adaptive Thresholding*

Tahap terakhir adalah *Adaptive Thresholding*, yaitu proses mengubah citra grayscale menjadi citra biner (hitam-putih). Metode ini menghitung nilai ambang secara lokal pada setiap area kecil citra (local thresholding), sehingga dapat menyesuaikan dengan kondisi pencahayaan yang tidak merata pada dokumen asli.

Kode implementasi *Adaptive Thresholding* dapat dilihat pada tabel berikut:

Tabel 4. 4 *Adaptive Thresholding*

```
# Adaptive Thresholding
block_size = max(3, self.block_size_slider["slider"].value() | 1)
c_value = self.c_slider["slider"].value()
self.processed_image = cv2.adaptiveThreshold(
    filtered, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN _C,
```

```
    cv2.THRESH_BINARY, block_size, c_value
)
```

Nilai parameter *Adaptive Thresholding* diatur melalui *slider* pada antarmuka aplikasi *preprocessing*. Nilai awal (*default*) untuk masing-masing parameter ditentukan pada saat pembuatan *slider*, sebagaimana ditunjukkan pada kode berikut.

Tabel 4. 5 Parameter Adaptive Threshold

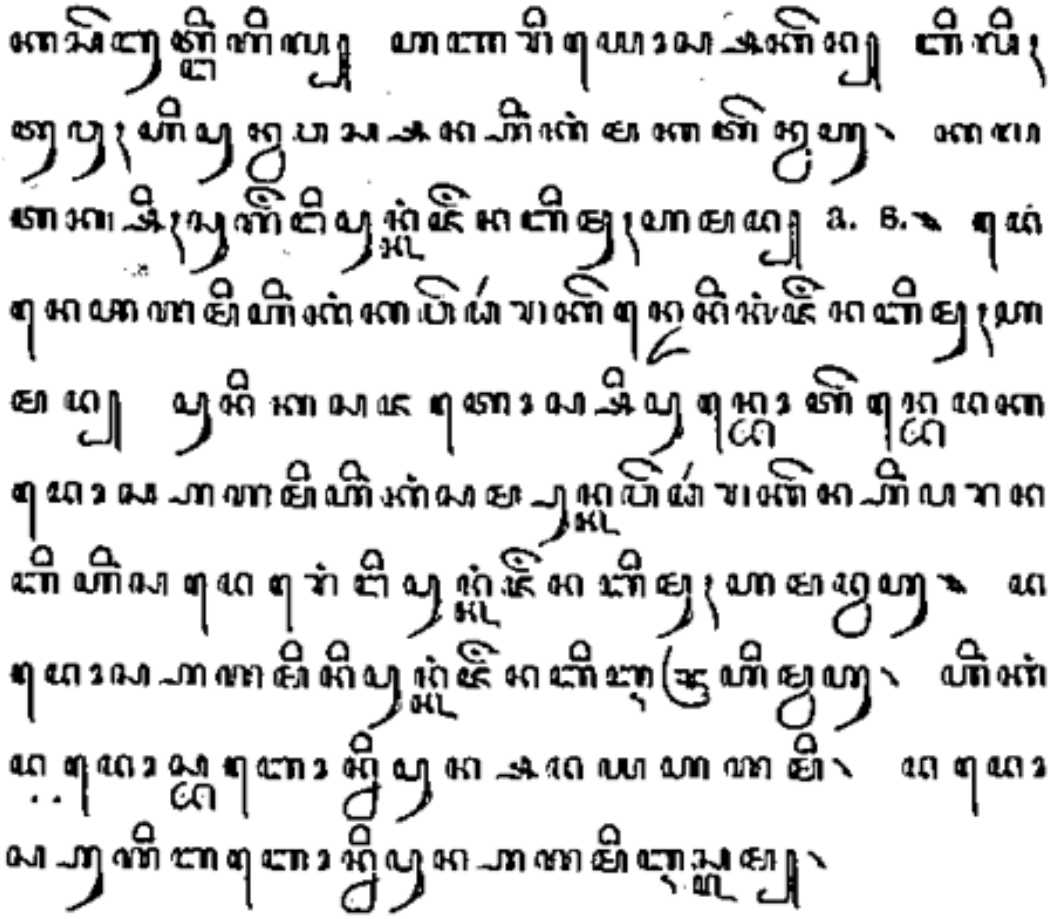
```
self.block_size_slider = self.create_slider(3, 255, 11, 2, "Block
Size:")
self.c_slider = self.create_slider(0, 50, 2, 1, "C Constant:")
```

Berdasarkan inisialisasi tersebut, nilai block size = 11 dan C = 2 digunakan sebagai konfigurasi awal dalam proses *Adaptive Thresholding*. Pemilihan nilai ini diperoleh melalui serangkaian percobaan berulang terhadap citra aksara Jawa dengan variasi kualitas yang berbeda.

Penggunaan nilai block size yang terlalu kecil menyebabkan hasil binerisasi menjadi sangat sensitif terhadap *noise*, sedangkan nilai C yang terlalu besar berpotensi menghilangkan bagian penting dari goresan aksara. Kombinasi block size = 11 dan C = 2 menghasilkan citra biner dengan tepi aksara yang lebih jelas serta latar belakang yang relatif bersih, sehingga lebih sesuai untuk tahap ekstraksi fitur HOG pada proses selanjutnya.

4.1.4 Hasil Akhir *Preprocessing*

Setelah ketiga tahap *preprocessing* diterapkan secara berurutan *Gaussian Blur*, *Bilateral Filter*, dan *Adaptive Thresholding* dihasilkan citra biner yang memperlihatkan bentuk aksara Jawa dengan jelas. Citra hasil *preprocessing* memperlihatkan kontur huruf yang tajam, bebas dari *noise*, dan kontras terhadap latar belakang. Hasil ini menjadi dasar untuk proses deteksi dan segmentasi aksara pada tahap selanjutnya.



Gambar 4. 1 Hasil *Preprocessing*

4.2 Deteksi dan Segmentasi Aksara

Tahapan ini bertujuan untuk menemukan posisi setiap aksara Jawa dalam citra hasil *preprocessing* dan kemudian memisahkan tiap aksara ke dalam file citra individual. Tahap ini terdiri dari dua bagian utama, yaitu deteksi (penentuan posisi huruf melalui bounding box) dan segmentasi (pemotongan huruf berdasarkan area deteksi).

4.2.1 Deteksi Aksara Jawa

Tahap deteksi dilakukan untuk menemukan posisi setiap karakter aksara Jawa pada citra hasil *preprocessing*. Metode ini menggunakan pendekatan deteksi kontur (contour detection) dari citra biner yang telah diolah sebelumnya. Tujuan utamanya adalah menghasilkan *bounding box* (kotak pembatas) di sekitar tiap karakter agar dapat dipisahkan secara otomatis pada tahap berikutnya.

1. Konversi dan Binerisasi Invers

Langkah awal adalah mengonversi citra berwarna menjadi grayscale, lalu menerapkan *Adaptive Thresholding* Inverse agar huruf tampak putih di atas latar belakang hitam. Pendekatan inverse thresholding ini digunakan karena OpenCV lebih optimal mendeteksi kontur pada objek berwarna terang di atas latar gelap.

Tabel 4. 6 Binerisasi Invers

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

thresh = cv2.adaptiveThreshold(

    gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN _C,

    cv2.THRESH_BINARY_INV, 15, 5

)
```

Parameter yang digunakan:

- Block size = 15 dan C = 5, diambil dari hasil uji coba pada citra hasil *preprocessing* yang memiliki variasi pencahayaan.
- Metode ADAPTIVE_THRESH_*GAUSSIAN* _C dipilih karena mampu menyesuaikan nilai ambang di setiap area lokal citra.

2. Operasi Morfologi Dilasi

Citra hasil thresholding masih memiliki bentuk huruf yang tipis, sehingga dilakukan dilasi morfologi untuk mempertebal garis dan menyatukan bagian huruf yang terputus. Dilasi ini menggunakan kernel berbentuk silang (cross-shaped kernel):

Tabel 4. 7 Operasi Dilasi

```
kernel = np.array([[0,1,0],
                   [1,1,1],
                   [0,1,0]], dtype=np.uint8)
dilated = cv2.dilate(thresh, kernel, iterations=1)
```

- Kernel 3×3 dipilih karena memberikan keseimbangan antara ketebalan garis dan kejelasan bentuk aksara.
- Iterasi = 1 digunakan agar huruf tidak saling menempel berlebihan.

3. Deteksi Kontur

Setelah citra diperjelas, sistem melakukan deteksi kontur menggunakan fungsi cv2.findContours(). Setiap kontur kemudian diubah menjadi koordinat persegi panjang menggunakan cv2.boundingRect(). Namun tidak semua kontur adalah huruf, sebagian merupakan *noise* atau tanda kecil. Karena itu, dilakukan penyaringan berdasarkan ukuran dan rasio aspek:

Tabel 4. 8 Deteksi Kontur

```
contours, _ = cv2.findContours(dilated, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)


bounding_boxes = []
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    if 8 < w < 60 and 10 < h < 80:
        aspect_ratio = w / h
        if 0.2 < aspect_ratio < 2.5:
            bounding_boxes.append((x, y, w, h))
```

Batas lebar (w) dan tinggi (h) diatur berdasarkan hasil pengamatan terhadap ukuran rata-rata aksara Jawa dari hasil digitalisasi. Huruf kecil (seperti ha, na) biasanya memiliki tinggi 30–60 piksel. Simbol atau pasangan cenderung lebih kecil dari 15 piksel, sehingga diabaikan. Rasio aspek 0.2–2.5 digunakan agar huruf yang terlalu pipih atau terlalu lebar tidak disertakan.

4. Penggabungan *Bounding box* Berdekatan

Pada beberapa kasus, huruf yang memiliki bagian menempel (misalnya pasangan ka dan ra) terdeteksi sebagai dua kotak terpisah. Untuk itu dilakukan penggabungan otomatis (merging) antar *bounding box* yang posisinya berdekatan dengan jarak toleransi 12 piksel.

Tabel 4. 9 Penggabungan Bounding Box

```
for i in range(len(bounding_boxes)):
    x, y, w, h = bounding_boxes[i]
    for j in range(len(merged_boxes)):
```

```
        mx, my, mw, mh = merged_boxes[j]
        if abs(mx - x) < 12 and abs(my - y) < 12:
            merged_boxes[j] = (
                min(mx, x), min(my, y),
                max(mx+mw, x+w) - min(mx, x),
                max(my+mh, y+h) - min(my, y)
            )
```

Teknik ini mengurangi kesalahan deteksi pada huruf majemuk dan menghasilkan kotak pembatas yang lebih representatif.

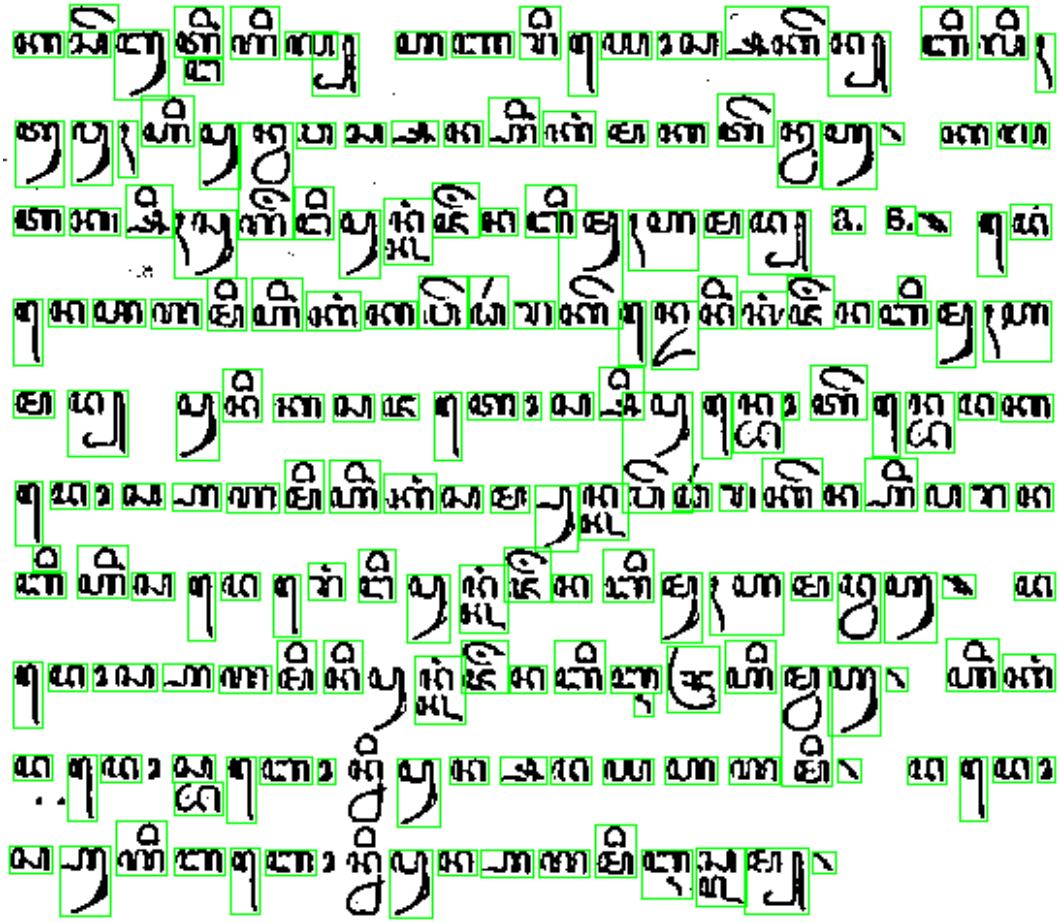
5. Visualisasi Bounding Box

Setiap kotak hasil akhir kemudian digambarkan di atas citra asli menggunakan garis hijau setipis mungkin agar tidak menutupi detail huruf.

Tabel 4. 10 Visualisasi Bounding Box

```
for x, y, w, h in merged_boxes:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 1)
```

Hasil deteksi ditunjukkan pada Gambar berikut:



Gambar 4. 2 Hasil Deteksi

4.2.2 Segmentasi Aksara Jawa

Tahap segmentasi dilakukan untuk memotong area setiap huruf berdasarkan posisi *bounding box* hasil deteksi pada tahap sebelumnya. Citra masukan berupa gambar dengan kotak hijau di sekitar huruf (hasil dari tahap deteksi). Tujuan utamanya adalah menghasilkan kumpulan file citra tunggal yang masing-masing berisi satu karakter aksara Jawa.

1. Deteksi Area *Bounding box* Berdasarkan Warna Hijau

Program mendeteksi area berwarna hijau sebagai penanda batas huruf. Rentang warna hijau ditentukan dengan batas bawah dan atas dalam format RGB agar dapat mengenali garis kotak yang digambar sebelumnya.

Tabel 4. 11 Deteksi Bounding Box

```
lower_green = np.array([0, 200, 0], dtype=np.uint8)
upper_green = np.array([100, 255, 100], dtype=np.uint8)
mask = cv2.inRange(self.image, lower_green, upper_green)
contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

Rentang warna ini dipilih karena garis *bounding box* digambar dengan nilai RGB sekitar (0, 255, 0), sehingga deteksi warna menjadi stabil meski ada variasi pencahayaan.

2. Pemotongan dan Penamaan Otomatis

Setiap kontur hasil deteksi hijau diubah menjadi koordinat kotak dan dipotong menggunakan indeks urutan. Hasil pemotongan disimpan otomatis ke folder tertentu dengan nama berurutan (crop_1.png, crop_2.png, dst).

Tabel 4. 12 Pemotongan Aksara

```
for idx, (x, y, w, h) in enumerate(bounding_boxes,
start=start_idx):
    cropped_image = self.image[y:y+h, x:x+w]
    output_path = os.path.join(save_path, f"crop_{idx}.png")
    cv2.imwrite(output_path, cropped_image)
```

Sistem juga memeriksa file yang sudah ada sebelumnya agar penomoran tetap berlanjut tanpa menimpa hasil lama. Hasil segmentasi ditunjukkan pada Gambar berikut.

Tabel 4. 13 Hasil Segmentasi

4.3 *Clustering*

Tahap *clustering* dilakukan melalui dua proses utama, yaitu ekstraksi fitur HOG dan pengelompokan menggunakan *Agglomerative Hierarchical Clustering* (AHC). Kedua proses ini bekerja berurutan untuk menghasilkan pengelompokan aksara berdasarkan kemiripan bentuk visualnya.

4.3.1 Ekstraksi Fitur HOG

Ekstraksi fitur dimulai dengan membaca setiap gambar hasil *Cropping*. Citra terlebih dahulu diubah ke dalam format grayscale karena metode HOG hanya memerlukan informasi intensitas untuk menghitung gradien.

Tabel 4. 14 Grayscale

```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Konversi ini bertujuan menyederhanakan informasi gambar agar fokus hanya pada struktur bentuk, bukan warna. Hal ini penting karena aksara Jawa

memiliki pola kurva dan garis dominan yang cukup jelas sehingga perhitungan gradien dalam grayscale sudah memadai.

Setelah konversi, gambar diubah ukurannya menjadi 64×64 piksel. Penyeragaman ukuran diperlukan agar seluruh citra menghasilkan panjang vektor fitur yang sama, sekaligus memastikan bahwa detail bentuk tetap cukup besar untuk dianalisis.

Tabel 4. 15 Resize

```
resized_image = cv2.resize(gray_image, (64, 64))
```

Ukuran 64×64 cukup untuk menampung struktur dasar aksara Jawa tanpa kehilangan karakteristik pentingnya, seperti lengkungan, garis tegak, atau titik kecil.

Tahap berikutnya adalah proses utama HOG. Metode HOG menghitung gradien pada setiap sel gambar, membuat histogram orientasi, dan menormalisasi hasilnya melalui blok. Parameter yang digunakan meliputi ukuran sel 8×8 piksel, ukuran blok 2×2 sel, dan sembilan orientasi gradien.

Tabel 4. 16 HOG

```
features, hog_image = hog(
    resized_image,
    pixels_per_cell=(8, 8),
    cells_per_block=(2, 2),
    orientations=9,
    visualize=True)
```

Proses ini menghasilkan vektor fitur berdimensi tinggi yang mewakili pola gradien pada gambar. Nilai orientasi dalam histogram mencerminkan arah garis dan

kurva pada aksara sehingga aksara yang memiliki bentuk mirip akan menghasilkan pola fitur yang relatif serupa. Setiap vektor fitur yang dihasilkan kemudian dikumpulkan dalam sebuah list untuk tahap *clustering*.

Tabel 4. 17 Pengumpulan Fitur

```
features_list.append(features)
```

Tahap ekstraksi fitur diakhiri dengan penyusunan seluruh fitur ke dalam sebuah array dua dimensi agar dapat diproses oleh algoritma *clustering*.

4.3.2 *Agglomerative Hierarchical Clustering*

Setelah seluruh fitur HOG terkumpul, tahap berikutnya adalah melakukan pengelompokan menggunakan metode *Agglomerative Hierarchical Clustering* (AHC). Metode ini berorientasi pada pendekatan bottom-up, yaitu dimulai dari unit data terkecil hingga membentuk cluster berukuran lebih besar. Dengan demikian, pada awal proses setiap citra aksara dianggap sebagai satu cluster independen.

Secara umum proses hierarkis dalam AHC berlangsung melalui tahapan berikut:

1. Inisialisasi Cluster Awal

Setiap gambar hasil ekstraksi fitur HOG diperlakukan sebagai satu cluster. Jika terdapat N gambar, maka terbentuk N cluster awal.

2. Perhitungan Jarak Antar Cluster

Sistem menghitung jarak antar fitur secara berpasangan untuk menentukan tingkat kemiripan antar cluster. Semakin kecil jarak antar fitur, semakin besar kemungkinan dua gambar memiliki kemiripan bentuk aksara.

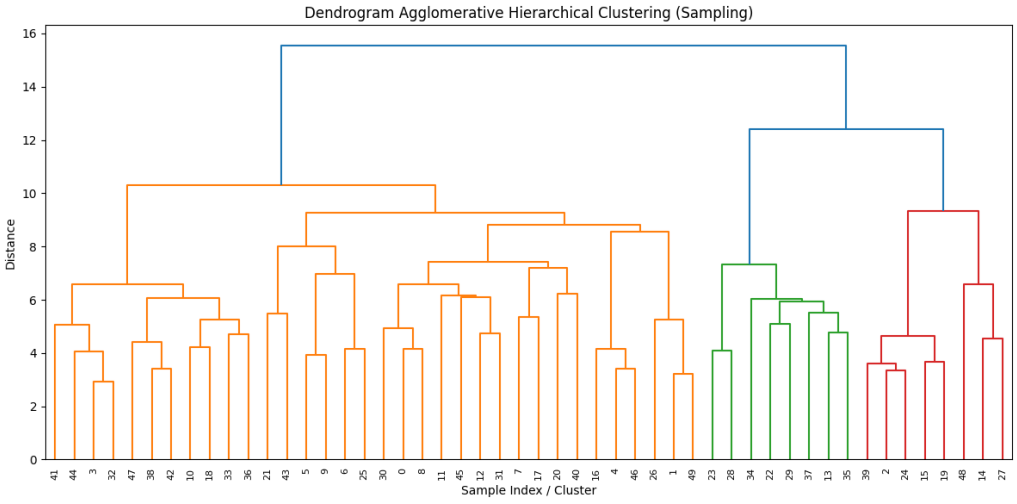
3. Penggabungan Dua Cluster Terdekat

Dua cluster dengan jarak terkecil digabung menjadi satu cluster baru. Setelah penggabungan terjadi, jarak antar cluster dihitung ulang karena struktur cluster telah berubah.

4. Iterasi Penggabungan hingga Pembentukan Struktur Hierarki

Proses penggabungan terus diulang hingga seluruh data tersusun dalam sebuah struktur hierarkis. Proses ini divisualisasikan dalam bentuk dendrogram, yang menunjukkan tahapan penyatuan cluster dari level paling kecil hingga level paling tinggi.

Untuk keperluan visualisasi, dataset berisi 9.844 citra memiliki kompleksitas komputasi sehingga dendrogram tidak dapat dihitung menggunakan seluruh data. Oleh karena itu, visualisasi dendrogram dibuat dengan random sampling 50 data representatif agar proses tetap ringan namun pola hierarki tetap terlihat. Dendrogram hasil proses hierarkis ditampilkan pada gambar berikut:



Gambar 4. 3 Proses Hierarkis

Melalui dendrogram terlihat bahwa proses penggabungan cluster berlangsung secara bertahap berdasarkan jarak kemiripan. Cluster yang mirip bergabung pada level hierarki rendah, sedangkan kelompok yang memiliki perbedaan bentuk aksara signifikan bergabung pada level hierarki yang lebih tinggi. Hal ini membuktikan bahwa pembentukan cluster dalam AHC tidak berjalan secara acak, tetapi mengikuti pola kemiripan struktural citra.

Setelah struktur hierarki terbentuk, sistem menentukan pemisahan cluster berdasarkan nilai distance_threshold agar jumlah cluster tidak ditentukan secara manual. Kode berikut menunjukkan konfigurasi model:

Tabel 4. 18 Pengelompokan AHC

```
clustering = AgglomerativeClustering(
    n_clusters=None,
    distance_threshold=5.0
)
clustering.fit(features_list)
```

Penggunaan distance_threshold sebesar 5.0 berarti proses penggabungan akan terus berlangsung selama jarak antar cluster masih berada di bawah ambang tersebut. Jika jarak antar cluster yang tersisa berada di atas nilai 5.0, proses hierarkis dihentikan dan jumlah cluster terakhir dianggap sebagai hasil akhir. Dengan pendekatan ini, jumlah cluster ditentukan secara otomatis oleh sistem berdasarkan kemiripan fitur, sehingga lebih fleksibel terhadap variasi bentuk aksara.

Setelah proses *clustering* selesai, setiap citra memperoleh label cluster sebagai identitas kelompok aksara yang mirip secara visual:

Tabel 4. 19 Pelabelan

```
cluster_labels = clustering.labels_
```

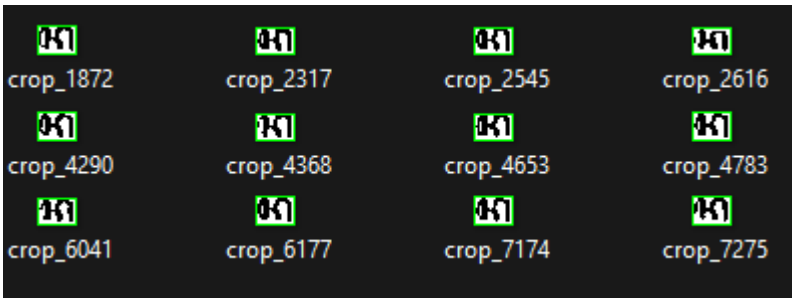
Label tersebut selanjutnya digunakan untuk mengelompokkan file gambar ke dalam folder sesuai cluster masing-masing. Sistem membuat folder seperti “Cluster_0”, “Cluster_1”, dan seterusnya, kemudian menyalin gambar ke dalam folder yang tepat.

Tabel 4. 20 Penyimpanan

```
destination_path = os.path.join(cluster_folder, image_name)

shutil.copy(image_path, destination_path)
```

Tahap *clustering* diakhiri dengan menyimpan data fitur dan label untuk keperluan evaluasi. Penyimpanan ini memungkinkan proses analisis kualitas cluster dilakukan pada tahap berikutnya.



Gambar 4. 4 Gambar Terkluser

4.4 Pembahasan

Tahap evaluasi *clustering* dilakukan untuk menilai kualitas pengelompokan aksara Jawa berdasarkan fitur HOG yang telah diekstraksi pada tahap sebelumnya. Evaluasi dilakukan menggunakan dua metrik utama, yaitu *Silhouette Score* dan *Davies Bouldin Index* (DBI). Kedua metrik ini dipilih karena mampu menggambarkan dua karakteristik penting dalam *clustering*, yaitu pemisahan antar

cluster dan kekompakan cluster. Pemilihan konfigurasi terbaik tidak dapat hanya mengandalkan salah satu metrik, sehingga penentuan dilakukan dengan mempertimbangkan keseimbangan antara keduanya.

Silhouette Score memberikan nilai antara -1 hingga 1, di mana nilai yang lebih tinggi menunjukkan bahwa cluster semakin terpisah. Sementara itu, Davies–Bouldin Index menghasilkan nilai ≥ 0 dan semakin rendah nilainya maka cluster semakin kompak dan stabil. Kombinasi kedua metrik memberikan gambaran menyeluruh mengenai kualitas *clustering* sehingga pengujian dilakukan pada beberapa variasi *distance threshold* pada metode *Agglomerative Hierarchical Clustering*. Hasil evaluasi *Silhouette Score* disajikan pada tabel berikut.

Tabel 4. 21 Hasil Silhoutte score

Pixels per cell	Cell per block	*Orientation*	*Distance threshold*	Jumlah Cluster Terbentuk	Silhoutte Score
8, 8	2, 2	9	5	1621	0,0564
8, 8	2, 2	9	6	838	0,05603
8, 8	2, 2	9	7	546	0,0585
8, 8	2, 2	9	8	381	0,0625
8, 8	2, 2	9	9	295	0,06507
8, 8	2, 2	9	10	237	0,0666
8, 8	2, 2	9	11	191	0,0763
8, 8	2, 2	9	12	159	0,0837

Berdasarkan hasil evaluasi *Silhouette Score* pada Tabel 4.21, terlihat bahwa nilai *Silhouette Score* cenderung meningkat seiring dengan bertambahnya *distance threshold*. Nilai tertinggi diperoleh pada threshold 12 dengan nilai 0,0837. Meskipun nilai ini tergolong rendah, tren peningkatan tersebut menunjukkan bahwa pada threshold yang lebih besar, pemisahan antar cluster menjadi relatif lebih baik. Hal ini disebabkan oleh proses penggabungan cluster pada *Agglomerative Hierarchical Clustering*, di mana cluster-cluster kecil yang sangat mirip secara fitur

digabung menjadi cluster yang lebih besar, sehingga jarak antar cluster yang tersisa menjadi lebih jelas.

Nilai *Silhouette Score* yang rendah secara keseluruhan disebabkan oleh karakteristik data aksara Jawa yang digunakan. Data berasal dari citra historis yang memiliki variasi bentuk tinggi, goresan yang tidak konsisten, aksara terputus, serta *noise* latar belakang. Kondisi ini menyebabkan representasi fitur HOG antar aksara sering kali tumpang tindih, sehingga batas antar cluster menjadi tidak tegas. Akibatnya, meskipun secara visual beberapa cluster tampak baik, secara numerik jarak antar cluster masih relatif dekat, yang tercermin pada nilai *Silhouette Score* yang rendah.

Tabel 4. 22 Hasil DBI

Pixels per cell	Cell per block	*Orientation*	*Distance threshold*	Jumlah Cluster Terbentuk	*Davies Bouldin Index*
8, 8	2, 2	9	5	1621	1,9434
8, 8	2, 2	9	6	838	2,4117
8, 8	2, 2	9	7	546	2,6666
8, 8	2, 2	9	8	381	2,7733
8, 8	2, 2	9	9	295	2,8211
8, 8	2, 2	9	10	237	2,904
8, 8	2, 2	9	11	191	2,8411
8, 8	2, 2	9	12	159	2,7621

Hasil evaluasi *Davies Bouldin Index* pada Tabel 4.22 menunjukkan bahwa nilai DBI terendah diperoleh pada threshold 5 dengan nilai 1,9434. Namun, pada konfigurasi ini jumlah cluster yang terbentuk sangat besar, yaitu 1.621 cluster.

Kondisi tersebut menunjukkan bahwa meskipun cluster-cluster yang terbentuk sangat kompak, struktur *clustering* menjadi terlalu terfragmentasi, sehingga banyak cluster yang memiliki citra dengan pola yang sama.

Nilai DBI cenderung meningkat ketika *distance threshold* diperbesar. Hal ini menunjukkan bahwa seiring penggabungan cluster menjadi semakin besar, tingkat kekompakan di dalam cluster mulai berkurang. Namun, nilai DBI tidak dapat dinilai secara terpisah tanpa melihat hasil pengelompokan secara visual. Pada threshold 12, meskipun nilai DBI sebesar 2,7621 bukan merupakan nilai terendah, jumlah cluster yang dihasilkan jauh lebih sedikit, yaitu 159 cluster. Kondisi ini membuat hasil pengelompokan menjadi lebih sederhana, lebih mudah dianalisis, dan lebih bermakna dalam merepresentasikan kelompok aksara. Dengan demikian, konfigurasi threshold 12 menunjukkan adanya keseimbangan antara kekompakan cluster dan tingkat generalisasi hasil pengelompokan.

Percobaan dibatasi hingga *distance threshold* 12 karena pada threshold di atas nilai tersebut, proses penggabungan cluster menghasilkan jumlah cluster yang semakin sedikit namun dengan kualitas visual yang menurun. Pengamatan visual menunjukkan bahwa pada threshold yang lebih besar, aksara dengan bentuk yang jelas berbeda mulai tergabung ke dalam cluster yang sama, sehingga mengurangi ketepatan dalam pengelompokan. Selain itu, kenaikan threshold di atas 12 tidak menunjukkan peningkatan signifikan pada nilai *Silhouette Score* maupun perbaikan struktur *clustering* secara visual, sehingga threshold 12 dipilih sebagai batas atas yang paling relevan untuk dianalisis lebih lanjut.

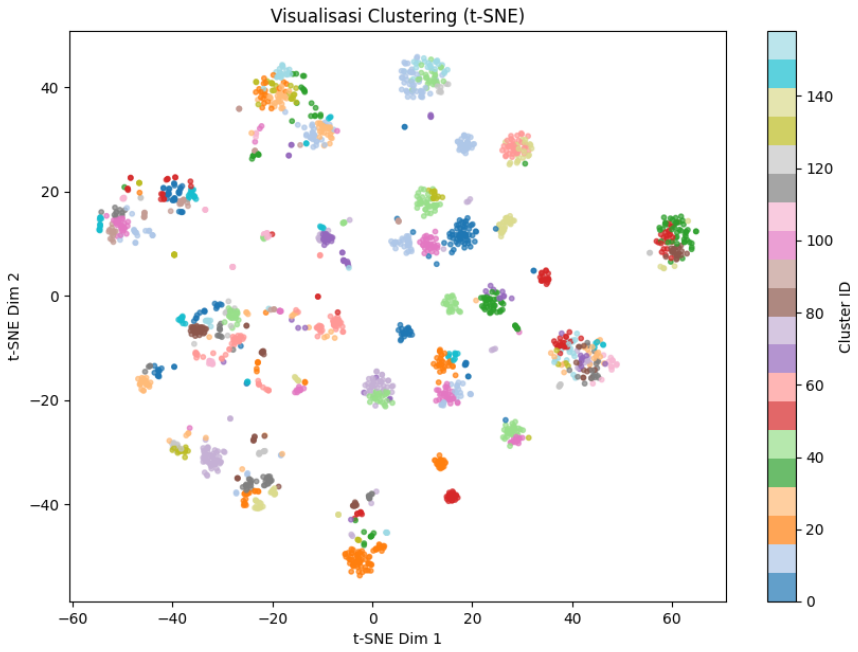
Dengan mempertimbangkan hasil evaluasi numerik menggunakan *Silhouette Score* dan Davies–Bouldin Index, jumlah cluster yang terbentuk, serta analisis visual hasil *clustering*, *distance threshold* 12 dipilih sebagai konfigurasi terbaik. Konfigurasi ini memberikan keseimbangan yang paling optimal antara pemisahan antar cluster, kekompakan cluster, dan interpretabilitas hasil pengelompokan aksara Jawa.

4.4.1 Visualisasi t-SNE

Visualisasi t-SNE digunakan untuk melihat struktur cluster dalam ruang berdimensi rendah. Visualisasi ini membantu mengonfirmasi apakah cluster yang terbentuk memiliki pemisahan yang jelas atau justru saling tumpang tindih.

1. Visualisasi t-SNE Konfigurasi Terbaik (*Distance threshold* = 12)

Visualisasi t-SNE untuk konfigurasi *distance threshold* 12 ditunjukkan pada Gambar 4.5 berikut.

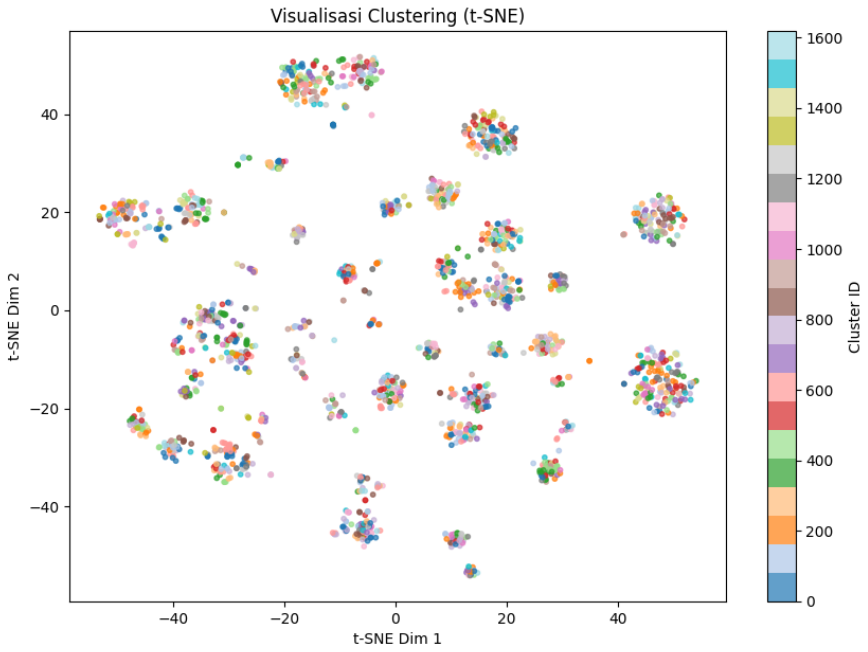


Gambar 4. 5 t-SNE 12

Pada konfigurasi ini, sebagian besar data membentuk kelompok-kelompok yang relatif padat dan memiliki jarak antar cluster yang cukup jelas. Meskipun masih terdapat beberapa area yang saling tumpang tindih, struktur pengelompokan secara umum lebih terorganisir. Kondisi ini sejalan dengan nilai *Silhouette Score* yang paling tinggi dan Davies–Bouldin Index yang masih berada pada rentang stabil, sehingga menunjukkan bahwa threshold 12 mampu memberikan keseimbangan antara pemisahan antar cluster dan kekompakan internal cluster.

2. Visualisasi t-SNE Konfigurasi Terburuk (*Distance threshold* = 5)

Visualisasi t-SNE untuk konfigurasi *distance threshold* 5 ditunjukkan pada Gambar 4.6 berikut.

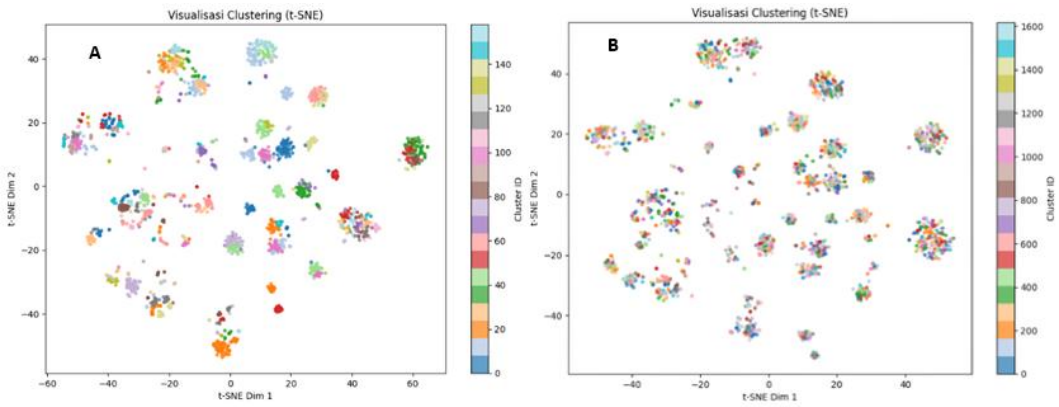


Gambar 4. 6 t-SNE 5

Berbeda dengan threshold 12, visualisasi pada threshold 5 menunjukkan sebaran titik yang lebih acak dan saling tumpang tindih. Jumlah cluster yang sangat banyak menyebabkan terbentuknya banyak kelompok kecil yang tidak memiliki struktur pemisahan yang jelas. Kondisi ini mengindikasikan bahwa meskipun nilai DBI pada threshold 5 tergolong rendah secara numerik, hasil *clustering* yang dihasilkan kurang bermakna secara visual dan sulit dianalisis.

3. Perbandingan Visualisasi t-SNE Threshold 12 dan Threshold 5

Perbandingan visualisasi t-SNE antara konfigurasi terbaik dan terburuk ditunjukkan pada Gambar 4.7, yang menampilkan kedua visualisasi secara sejajar. Label A menunjukkan t-SNE pada *distance threshold* 12, sedangkan label B menunjukkan t-SNE pada *distance threshold* 5.



Gambar 4. 7 Perbandingan Visual

Pada visualisasi gabungan tersebut terlihat secara jelas perbedaan struktur *clustering* yang terbentuk. Visualisasi A (threshold 12) menunjukkan cluster yang lebih terkelompok dan terpisah, sedangkan visualisasi B (threshold 5) memperlihatkan pola sebaran yang tidak terstruktur dan saling tumpang tindih. Perbandingan ini memperkuat hasil evaluasi numerik bahwa *distance threshold* 12 merupakan konfigurasi yang paling optimal dalam pengelompokan aksara Jawa, sementara threshold 5 menjadi konfigurasi yang paling buruk.

4.4.2 Temuan

Berdasarkan hasil evaluasi numerik dan visual *clustering*, terdapat beberapa temuan penting yang memberikan gambaran mengenai kinerja algoritma *Agglomerative Hierarchical Clustering* dalam mengelompokkan citra aksara Jawa berbasis fitur HOG.

Secara keseluruhan, nilai *Silhouette Score* dan Davies–Bouldin Index yang diperoleh masih tergolong rendah. Hal ini mengindikasikan bahwa pemisahan antar cluster belum benar-benar jelas dan kekompakan dalam cluster masih dapat ditingkatkan. Kondisi ini dapat dipahami mengingat karakteristik aksara Jawa

memiliki variasi bentuk yang sangat tinggi, ukuran goresan yang tidak seragam, serta ketergantungan pada detail kurva yang relatif halus. Selain itu, sebagian citra hasil segmentasi mengalami ketidakstabilan visual seperti goresan tipis, bagian yang terputus, atau perbedaan ketebalan tinta. Faktor-faktor tersebut menyebabkan ekstraksi fitur HOG menghasilkan representasi visual yang mirip untuk bentuk aksara yang berbeda maupun sebaliknya.

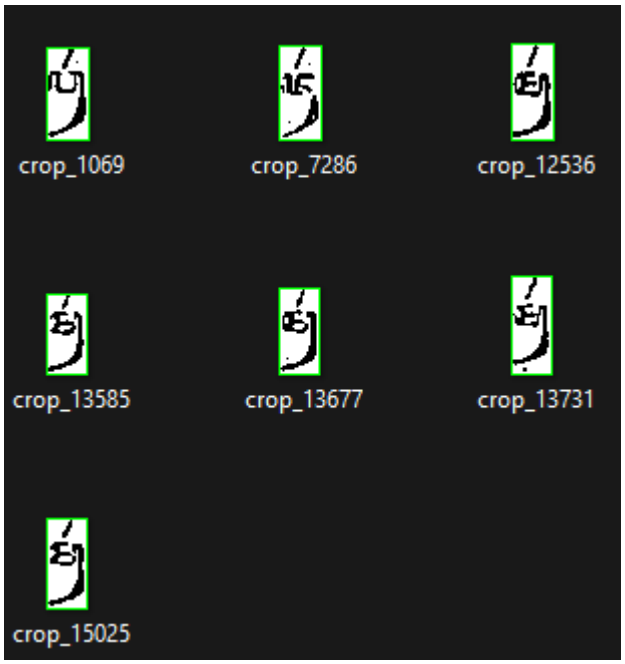
Pada konfigurasi *distance threshold* 5, yang merupakan konfigurasi terburuk. Pada konfigurasi ini, banyak aksara yang seharusnya berada dalam satu cluster justru terpisah ke berbagai cluster yang berbeda seperti yang ditunjukkan pada gambar 4.7 dan 4.8 berikut.

Gambar 4. 8 DT 5 Cluster 22

Gambar 4. 9 DT5 Cluster 25

Hal tersebut terjadi karena jumlah cluster yang terbentuk mencapai 1621, sehingga setiap variasi minor dalam bentuk aksara dianggap berbeda secara signifikan oleh algoritma. Dengan kata lain, cluster terlalu terfragmentasi sehingga tidak lagi merepresentasikan kesamaan bentuk aksara secara bermakna.

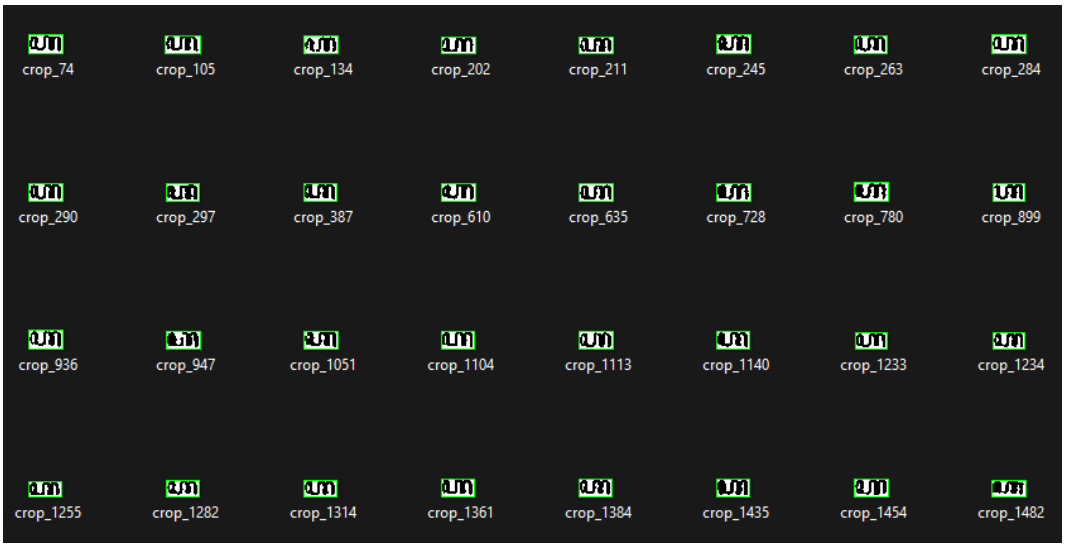
Pada konfigurasi *distance threshold* 5 juga ditemukan pola yang berlawanan, yaitu pada beberapa kasus terdapat aksara yang seharusnya berada pada cluster yang berbeda karena perbedaan bentuk visual yang jelas, namun justru terkelompok ke dalam cluster yang sama.



Gambar 4. 10 DT 5 Cluster 1

Kondisi ini disebabkan oleh adanya kemiripan pola gradien pada citra yang secara struktur geometri cukup berbeda, namun secara tekstur lokal terbaca mirip oleh HOG sehingga algoritma menganggapnya sebagai satu kategori.

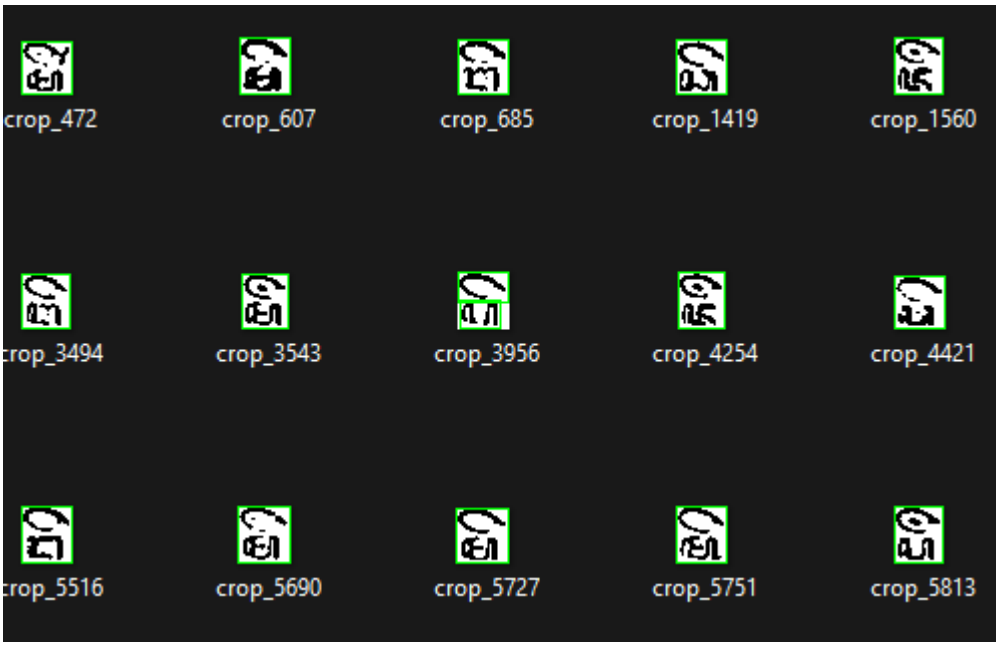
Sementara itu, konfigurasi threshold 12, yang merupakan konfigurasi terbaik, menunjukkan pemisahan cluster yang jauh lebih jelas secara visual. Pada banyak kelompok, aksara yang berada dalam cluster yang sama benar-benar memperlihatkan kemiripan bentuk yang kuat.



Gambar 4. 11 DT 12 Cluster 1

Hal ini memperkuat kesimpulan bahwa threshold 12 secara umum memberikan struktur pengelompokan paling stabil dan bermakna.

Namun demikian, bahkan pada threshold terbaik masih ditemukan beberapa kasus di mana aksara yang seharusnya berbeda berada dalam cluster yang sama, meskipun tingkat kesalahannya jauh lebih kecil dibanding threshold 5.



Gambar 4. 12 DT 12 Cluster 5

Setelah diamati lebih dekat, kesalahan tersebut dapat dijelaskan oleh beberapa faktor: goresan aksara yang tidak jelas, bagian huruf yang terputus, citra terlalu tipis, atau pola HOG yang terbaca terlalu mirip akibat bentuk visual aksara yang hampir sama. Dengan kata lain, sebagian kesalahan bukan sepenuhnya berasal dari algoritma *clustering*, melainkan dari kualitas citra aksara yang tidak konsisten karena sifat historis.

Dari seluruh temuan tersebut dapat disimpulkan bahwa kinerja *clustering* tidak hanya dipengaruhi oleh konfigurasi parameter algoritma, tetapi juga kualitas input visual dan konsistensi representasi fitur. Meskipun nilai metrik evaluasi tidak mencapai kategori tinggi, struktur cluster yang diperoleh pada threshold 12 sudah cukup stabil dan representatif untuk tujuan pemetaan kemiripan bentuk aksara Jawa.

4.5 Integrasi Islam

Integrasi nilai-nilai Islam dalam penelitian ini didasarkan pada dua landasan utama, yaitu perintah mengembangkan ilmu pengetahuan dan anjuran untuk memahami serta menghargai keragaman bahasa. Kedua prinsip ini menjadi dasar yang melandasi kegiatan ilmiah dalam konteks apa pun, termasuk penelitian mengenai aksara Jawa menggunakan teknologi modern.

4.5.1 Perintah Iqra’ dan Pengembangan Ilmu Pengetahuan

Fondasi utama pengembangan ilmu dalam Islam berangkat dari wahyu pertama yang diturunkan kepada Nabi Muhammad SAW. Ayat tersebut tidak hanya

memerintahkan aktivitas membaca, tetapi juga menekankan pentingnya proses belajar, penelitian, dan pengembangan pengetahuan. Ayat tersebut berbunyi:

اِقْرَأْ بِاسْمِ رَبِّكَ الَّذِيْ خَلَقَۚ ١ خَلَقَ الْاِنْسَانَ مِنْ عَلَقٍۚ ٢ اِقْرَأْ وَرَبُّكَ الْاَكْرَمُۙ ٣ الَّذِيْ عَلَّمَ بِالْقَلَمِۙ ٤ عَلَّمَ الْاِنْسَانَ مَا لَمْ يَعْلَمْۗ ٥

"Bacalah dengan (menyebut) nama Tuhanmu yang menciptakan, Dia menciptakan manusia dari segumpal darah. Bacalah, dan Tuhanmulah Yang Maha Pemurah, yang mengajar manusia dengan pena. Dia mengajarkan manusia apa yang tidak diketahuinya." (QS. Al-'Alaq ayat 1–5)

Dalam Tafsir Ibn Katsir (Ibn Kathir, 2016), ayat ini menegaskan beberapa poin:

- Iqra' tidak terbatas pada membaca teks, tetapi mencakup belajar, merenung, meneliti, dan memahami fenomena.
- Penyebutan pena (al-qalam) menunjukkan pentingnya pencatatan dan dokumentasi ilmu sebagai metode perpindahan pengetahuan antargenerasi.
- Allah adalah sumber segala ilmu, sehingga setiap aktivitas ilmiah memiliki nilai spiritual apabila dilakukan dengan niat memperoleh pemahaman yang benar.

Hadis sahih berikut turut memperkuat dorongan mencari ilmu:

"Barang siapa menempuh jalan untuk mencari ilmu, Allah akan memudahkan baginya jalan menuju surga." (HR. Muslim no. 2699)

Para ulama seperti Imam Al-Ghazali dalam Ihya' Ulumuddin menjelaskan bahwa ilmu yang bermanfaat ('ilm nafi') adalah ilmu yang diperoleh melalui metode yang benar, penuh ketelitian, dan memberikan manfaat bagi manusia. Dengan demikian, setiap penelitian ilmiah, termasuk di bidang bahasa dan

teknologi, berakar pada dorongan syariat untuk terus mengembangkan pengetahuan secara sistematis dan bertanggung jawab (Al-Ghazali, 2011).

4.5.2 Keragaman Bahasa sebagai Tanda Kebesaran Allah

Bahasa dalam Islam dipandang sebagai salah satu tanda kebesaran Allah yang menunjukkan keluasan ciptaan-Nya. Keragaman bahasa manusia bukan hanya fenomena sosial, tetapi juga memiliki dimensi spiritual sebagai bagian dari ayat kauniyah. Hal ini dijelaskan dalam firman Allah:

وَمِنْ اٰيٰتِهٖ خَلْقُ السَّمٰوٰتِ وَالْاَرْضِ وَاخْتِلَافُ اَلْسِنَتِكُمْ وَاَلْوَانِكُمْۗ اِنَّ فِيْ ذٰلِكَ لَاٰيٰتٍ لِّلْعٰلِمِيْنَ ٢٢

"*Dan* di antara tanda-tanda (kekuasaan)-Nya ialah penciptaan langit dan bumi dan perbedaan bahasa-bahasamu dan warna kulitmu. Sesungguhnya pada yang demikian itu benar-benar terdapat tanda-tanda bagi orang-orang yang mengetahui." (QS. Ar-Rum ayat 22)

Dalam Tafsir Al-Tabari (At-Tabari, n.d.), ayat ini mengandung beberapa poin penting:

- Keragaman bahasa adalah bukti nyata kekuasaan Allah dan bukan hal yang tercipta secara kebetulan.
- Bahasa merupakan sarana utama bagi manusia untuk berpikir, memahami, dan mewariskan ilmu.
- Orang yang mempelajari bahasa serta merenungi keragamannya termasuk golongan orang-orang yang berilmu (ulul albab), karena mereka memahami hikmah di balik fenomena tersebut.

Tokoh pemikiran modern seperti Syed Muhammad Naquib al-Attas juga menjelaskan bahwa bahasa adalah wadah makna. Jika suatu bahasa atau

ditinggalkan, maka sebagian pengetahuan dan cara pandang masyarakat tersebut ikut hilang (Ahmad, 2019). Dalam konteks ini, upaya mempelajari dan menjaga aksara Jawa melalui dokumentasi, digitalisasi, dan pengolahan data menjadi bagian dari penghargaan terhadap keragaman bahasa yang Allah ciptakan. Pelestarian ini memungkinkan pengetahuan yang tersimpan di masa lalu tetap dapat diakses, dipelajari, dan dikembangkan oleh generasi berikutnya.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa pengelompokan aksara Jawa dapat dilakukan dengan memanfaatkan fitur *Histogram of Oriented Gradients* (HOG) dan metode *Agglomerative Hierarchical Clustering*. Evaluasi kualitas *clustering* menggunakan *Silhouette Score* dan Davies–Bouldin Index menunjukkan bahwa nilai *distance threshold* 12 memberikan hasil terbaik dibandingkan konfigurasi lainnya, dengan *Silhouette Score* sebesar 0,0837 dan Davies–Bouldin Index sebesar 2,7621, serta menghasilkan 159 cluster. Konfigurasi ini dinilai paling optimal karena mampu memberikan pemisahan antar cluster yang relatif baik sekaligus menjaga tingkat kekompakan cluster yang stabil. Meskipun demikian, nilai evaluasi yang diperoleh masih tergolong rendah secara keseluruhan, yang dapat diakibatkan oleh pengaruh kualitas citra aksara historis seperti goresan yang tidak jelas, aksara terputus, serta variasi bentuk yang tinggi, sehingga memengaruhi konsistensi representasi fitur dan hasil pengelompokan.

5.2 Saran

Berdasarkan temuan dan kendala yang dijumpai selama penelitian, beberapa saran yang dapat menjadi rujukan untuk peningkatan dan pengembangan penelitian selanjutnya adalah sebagai berikut:

1. Penerapan teknik peningkatan kualitas citra tingkat lanjut (image restoration / super-resolution) direkomendasikan untuk mengatasi ketidakjelasan goresan aksara, bagian huruf yang terputus, dan ketebalan tinta yang tidak konsisten pada dokumen historis.
2. Eksplorasi metode ekstraksi fitur yang lebih kuat seperti SIFT, LBP, SURF, atau pembelajaran berbasis CNN penting dilakukan untuk membandingkan performa representasi fitur terhadap HOG.
3. Penggunaan teknik *clustering* alternatif seperti DBSCAN dapat dievaluasi untuk melihat apakah struktur pengelompokan yang dihasilkan lebih stabil dibanding *Agglomerative Hierarchical Clustering*.
4. Integrasi hasil *clustering* ke tahap labeling dataset.

DAFTAR PUSTAKA

Afakh, M. L., Risnumawan, A., & Anggraeni, M. E. A. (2018). Aksara Jawa Text Detection in Scene Images using Convolutional Neural Network.

Ahmad, S. (2019). Al-Attas on language and thought: Its relation to worldview, change and translation. TAFHIM: IKIM Journal of Islam and the Contemporary World, 12(2). https://doi.org/10.56389/tafhim.vol12no2.4

Ah-Pine, J. (2018). An efficient and effective generic *Agglomerative Hierarchical Clustering* approach. Journal of Machine Learning Research, 19, 1–43.

Al-Ghazālī. (2011). Iḥyā' 'Ulūm al-Dīn (A. A. Mas'adi, Trans.). Jakarta: Republika.

At-Tabari. (n.d.). Jami' al-Bayan fi Ta'wil al-Qur'an (Terj. Bahasa Indonesia; Tahqiq: Ahmad Abdurraziq Al-Bakri, Muhammad Adil Muhammad, Muhammad Abdul Lathif Khalaf, & Mahmud Mursi Abdul Hamid; Penyunting: Ahmad Muhammad Syakir & Mahmud Muhammad Syakir). Pustaka Azzam. Retrieved from https://archive.org/details/tafsir-1_202201/Tafsir%2020

Burney, S. M. A., & Tariq, H. (2015). K-Means Cluster Analysis for Image Segmentation.

Cai, P., Cai, Z., Fan, Y., Zhang, Y., & Liu, Y. (2024). Image contour detection based on visual pathway information transfer mechanism. Neural Processing Letters, 56(6).

Damayanti, F., Kusnendar Suprapto, Y., & Yuniarno, E. M. (2019). Segmentation of Javanese Characters in Ancient Manuscript using Connected Component Labeling.

Deore, S. P., & Pravin, A. (2019). *Histogram of Oriented Gradients* based off-line handwritten Devanagari characters recognition using SVM, K-NN and NN classifiers. Revue d'Intelligence Artificielle, 33(6), 441–446. https://doi.org/10.18280/ria.330606

Fahim, A. M., Salem, A.-B. M., & Ramadan, M. A. (2006). An efficient enhanced k-means *clustering* algorithm.

Fakhruddin, M., Sachari, A., & Haswanto, N. (2023). Pengembangan Desain Informasi dan Pembelajaran Aksara Jawa melalui Media Website.

Hartama, D., & Anjelita, M. (2022). Analysis of Silhouette Coefficient Evaluation with Euclidean Distance in the *Clustering* Method (Case Study: Number of Public Schools in Indonesia). Jurnal Mantik, 6(3), 2941–2947. Retrieved from https://iocscience.org/ejournal/index.php/mantik/article/view/3318

Himamunanto, A. R., & Widiarti, A. R. (2017). Javanese Character Image Segmentation of Document Image of Hamong Tani.

Hussain, S. F., & Bashir, S. (2016). Co-*clustering* of multi-view datasets. Knowledge and Information Systems.

Ibn Kathir. (2016). Tafsir al-Qur'an al-'Azim (M. Abdul Ghaffar, Trans.). Jakarta: Pustaka Imam Asy-Syafi'i.

Idrus, A., Tarihoran, N., Supriatna, U., Tohir, A., Suwarni, S., & Rahim, R. (2022). Distance analysis measuring for *clustering* using K-Means and Davies–Bouldin Index algorithm. TEM Journal, 11(4), 1871–1876. https://doi.org/10.18421/TEM114-55

Jardim, S., António, J., & Mora, C. (2022). Graphical image region extraction with K-means *clustering* and watershed.

Joseph, J., & Periyasamy, R. (2018). An image driven *Bilateral Filter* with adaptive range and spatial parameters for denoising magnetic resonance images. Computers & Electrical Engineering, 69, 782–795. https://doi.org/10.1016/j.compeleceng.2018.02.033

Khan, N. H., Adnan, A., & Basar, S. (2018). Urdu ligature recognition using multi-level *Agglomerative Hierarchical Clustering*. Cluster Computing, 21(1), 503–514. https://doi.org/10.1007/s10586-017-0916-2

Kowalczyk, P., Izydorczyk, J., & Szelest, M. (2022). Evaluation Methodology for Object Detection and Tracking in *Bounding box* Based Perception Modules. Electronics, 11(8), 1182. https://doi.org/10.3390/electronics11081182

Moëllic, P.-A., Haugeard, J.-E., & Pittel, G. (2018). Image *Clustering* Based on a Shared Nearest Neighbors Approach for Tagged Collections.

Oti, E. U., & Olusola, M. O. (2024). Overview of *Agglomerative Hierarchical Clustering* methods. British Journal of Computer, Networking and Information Technology, 7(2), 14–23. https://doi.org/10.52589/BJCNIT-CV9POOGW

Peikari, M., Salama, S., Nofech-Mozes, S., et al. (2018). A cluster-then-label semi-supervised learning approach for pathology image classification. Scientific Reports, 8, 7193. https://doi.org/10.1038/s41598-018-24876-0

Said, K., & Jambek, A. (2021). Analysis of image processing using morphological erosion and dilation. Journal of Physics: Conference Series, 2071(1), 012033.

Sehgal, I., & Venkatesh, K. S. (2019). Connected component labeling for binary images. International Journal of Advanced Research.

Smolka, B., Kusnik, D., & Radlak, K. (2023). On the reduction of mixed *Gaussian* and impulsive *noise* in heavily corrupted color images. Scientific Reports, 13, 21035.

Sugianela, M., & Suciati, N. (2019). Javanese document image recognition using multiclass support vector machine. In Proceedings of the 2019 International Conference on Information and Communications Technology (ICOIACT) (pp. 420–424). IEEE. https://doi.org/10.1109/ICOIACT46704.2019.8938546

Tanaya, D., & Adriani, M. (2018). Word Segmentation for Javanese Character using Dictionary, SVM, and CRF.

Wang, M., Li, J., Su, H., Yin, N., Yang, L., & Li, S. (2024). GraphCL: Graph-based *clustering* for semi-supervised medical image segmentation.

LAMPIRAN

Preprocessing

```
import cv2
import numpy as np
import sys
import subprocess
import os
from PyQt6.QtWidgets import (
    QApplication, QWidget, QLabel, QPushButton, QFileDialog,
    QSlider, QVBoxLayout, QHBoxLayout, QScrollArea, QComboBox
)
from PyQt6.QtCore import Qt
from PyQt6.QtGui import QPixmap, QImage

class ImageProcessor(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Adaptive Thresholding with
Preprocessing")
        self.setGeometry(100, 100, 1000, 600)

        # Variabel gambar
        self.image = None
        self.processed_image = None

        # Scroll Area untuk gambar
        self.scroll_area = QScrollArea(self)
        self.image_label = QLabel()
        self.image_label.setAlignment(Qt.AlignmentFlag.AlignCent
er)
        self.scroll_area.setWidget(self.image_label)
        self.scroll_area.setWidgetResizable(True)
        self.scroll_area.setFixedSize(600, 500)

        # Tombol utama
        self.open_button = QPushButton("Open Image")
        self.save_button = QPushButton("Save Image")

        # Dropdown untuk memilih mode (Bounding Box / Cropping)
        self.mode_selector = QComboBox()
        self.mode_selector.addItem("Preprocessing")
        self.mode_selector.addItem("Bounding Box")
        self.mode_selector.addItem("Cropping")

        # Slider untuk thresholding dan filter
```

```
        self.gaussian_slider = self.create_slider(1, 11, 3, 2,
"Gaussian Blur:")
        self.bilateral_diameter_slider = self.create_slider(1,
25, 15, 2, "Bilateral Diameter:")
        self.bilateral_sigma_slider = self.create_slider(10,
200, 75, 5, "Bilateral Sigma:")
        self.block_size_slider = self.create_slider(3, 255, 11,
2, "Block Size:")
        self.c_slider = self.create_slider(0, 50, 2, 1, "C
Constant:")

        # Layout kiri (gambar)
        left_layout = QVBoxLayout()
        left_layout.addWidget(self.scroll_area,
alignment=Qt.AlignmentFlag.AlignCenter)

        # Layout kanan (slider & tombol)
        right_layout = QVBoxLayout()
        right_layout.addWidget(self.open_button)
        right_layout.addWidget(self.save_button)
        right_layout.addWidget(self.mode_selector)

        # 1. Gaussian Blur
        right_layout.addLayout(self.gaussian_slider["layout"])

        # 2. Bilateral Filter
        right_layout.addLayout(self.bilateral_diameter_slider["l
ayout"])
        right_layout.addLayout(self.bilateral_sigma_slider["layo
ut"])

        # 3. Adaptive Thresholding
        right_layout.addLayout(self.block_size_slider["layout"])
        right_layout.addLayout(self.c_slider["layout"])

        right_layout.addStretch()


        # Layout utama (kiri-kanan)
        main_layout = QHBoxLayout()
        main_layout.addLayout(left_layout, 2)
        main_layout.addLayout(right_layout, 1)

        self.setLayout(main_layout)
```

```
        # Event handling
        self.open_button.clicked.connect(self.open_image)
        self.save_button.clicked.connect(self.save_image)
        self.mode_selector.currentIndexChanged.connect(self.hand
le_mode_change)

        self.block_size_slider["slider"].valueChanged.connect(se
lf.apply_threshold)
        self.c_slider["slider"].valueChanged.connect(self.apply_
threshold)
        self.gaussian_slider["slider"].valueChanged.connect(self
.apply_threshold)
        self.bilateral_diameter_slider["slider"].valueChanged.co
nnect(self.apply_threshold)
        self.bilateral_sigma_slider["slider"].valueChanged.conne
ct(self.apply_threshold)

    def create_slider(self, min_val, max_val, default, step,
label_text):
        """Membuat slider dengan label"""
        layout = QVBoxLayout()
        label = QLabel(f"{label_text} {default}")
        slider = QSlider(Qt.Orientation.Horizontal)
        slider.setMinimum(min_val)
        slider.setMaximum(max_val)
        slider.setValue(default)
        slider.setSingleStep(step)
        layout.addWidget(label)
        layout.addWidget(slider)

        def update_label(value):
            """Update nilai slider di label"""
            if "Gaussian Blur" in label_text:
                value = max(1, value | 1)
                slider.setValue(value)
            label.setText(f"{label_text} {value}")

        slider.valueChanged.connect(update_label)
        return {"slider": slider, "layout": layout}

    def open_image(self):
        """Membuka dialog untuk memilih gambar"""
        file_path, _ = QFileDialog.getOpenFileName(self, "Open
Image", "", "Images (*.png *.jpg *.jpeg)")
        if file_path:
```

```
            self.image = cv2.imread(file_path,
cv2.IMREAD_GRAYSCALE)
            self.apply_threshold()

    def apply_threshold(self):
        """Menerapkan preprocessing dan adaptive thresholding"""
        if self.image is None:
            return

        # **1. Gaussian Blur**
        kernel_size = max(1,
self.gaussian_slider["slider"].value() | 1)
        blurred = cv2.GaussianBlur(self.image, (kernel_size,
kernel_size), 0)

        # **2. Bilateral Filter**
        d = self.bilateral_diameter_slider["slider"].value()
        sigma = self.bilateral_sigma_slider["slider"].value()
        filtered = cv2.bilateralFilter(blurred, d=d,
sigmaColor=sigma, sigmaSpace=sigma)

        # **3. Adaptive Thresholding**
        block_size = max(3,
self.block_size_slider["slider"].value() | 1)
        c_value = self.c_slider["slider"].value()
        self.processed_image = cv2.adaptiveThreshold(
            filtered, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
            cv2.THRESH_BINARY, block_size, c_value
        )

        self.display_image(self.processed_image)

    def display_image(self, img):
        """Menampilkan gambar di QLabel dalam ukuran aslinya"""
        h, w = img.shape
        bytes_per_line = w
        q_image = QImage(img.data, w, h, bytes_per_line,
QImage.Format.Format_Grayscale8)

        pixmap = QPixmap.fromImage(q_image)
        self.image_label.setPixmap(pixmap)
        self.image_label.resize(pixmap.size())

    def save_image(self):
```

```
        """Menyimpan gambar hasil processing dengan nama auto-
urut"""
        if self.processed_image is None:
            return

        save_path = "E:/codefix/gambarpreproses2/"
        if not os.path.exists(save_path):
            os.makedirs(save_path)

        existing_files = [f for f in os.listdir(save_path) if
f.startswith("processed_") and f.endswith(".png")]
        existing_numbers = [
            int(f.replace("processed_", "").replace(".png", ""))
for f in existing_files if f.replace("processed_",
"").replace(".png", "").isdigit()
        ]

        next_number = max(existing_numbers) + 1 if
existing_numbers else 1
        file_name = f"processed_{next_number}.png"
        file_path = os.path.join(save_path, file_name)

        cv2.imwrite(file_path, self.processed_image)
        print(f"☑ Gambar berhasil disimpan di: {file_path}")

    def handle_mode_change(self):
        """Menjalankan script berdasarkan pilihan di dropdown
dan menutup GUI utama"""
        mode = self.mode_selector.currentText()

        if mode == "Preprocessing":
            return

        script_path = os.path.join(os.path.dirname(__file__),
f"{mode.lower().replace(' ', '')}.py")

        if os.path.exists(script_path):
            subprocess.Popen(["python", script_path])
            self.close()
        else:
            print(f"❌ {script_path} tidak ditemukan!")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = ImageProcessor()
```

```
    window.show()
    sys.exit(app.exec())
```

Bounding Box

```
import cv2
import numpy as np
import sys
import os
import subprocess
from PyQt6.QtWidgets import (
    QApplication, QWidget, QLabel, QPushButton, QFileDialog,
    QScrollArea, QVBoxLayout, QHBoxLayout, QComboBox
)
from PyQt6.QtCore import Qt
from PyQt6.QtGui import QPixmap, QImage

class BoundingBoxApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Bounding Box Detector")
        self.setGeometry(100, 100, 1000, 600)

        # Variabel gambar
        self.image = None
        self.processed_image = None

        # Scroll Area untuk gambar
        self.scroll_area = QScrollArea(self)
        self.image_label = QLabel()
        self.image_label.setAlignment(Qt.AlignmentFlag.AlignCent
er)
        self.scroll_area.setWidget(self.image_label)
        self.scroll_area.setWidgetResizable(True)
        self.scroll_area.setFixedSize(600, 500)

        # Tombol utama
        self.open_button = QPushButton("Open Image")
        self.save_button = QPushButton("Save Image")
        self.save_button.setEnabled(False)

        # Dropdown untuk memilih mode (Prepro2 / Cropping)
        self.mode_selector = QComboBox()
```

```
        self.mode_selector.addItem("Bounding Box")
        self.mode_selector.addItem("Preprocessing")
        self.mode_selector.addItem("Cropping")

        # Layout kiri (gambar)
        left_layout = QVBoxLayout()
        left_layout.addWidget(self.scroll_area,
alignment=Qt.AlignmentFlag.AlignCenter)

        # Layout kanan (tombol & dropdown)
        right_layout = QVBoxLayout()
        right_layout.addWidget(self.open_button)
        right_layout.addWidget(self.save_button)
        right_layout.addWidget(self.mode_selector)
        right_layout.addStretch()

        # Layout utama (kiri-kanan)
        main_layout = QHBoxLayout()
        main_layout.addLayout(left_layout, 2)
        main_layout.addLayout(right_layout, 1)

        self.setLayout(main_layout)

        # Event handling
        self.open_button.clicked.connect(self.open_image)
        self.save_button.clicked.connect(self.save_image)
        self.mode_selector.currentIndexChanged.connect(self.hand
le_mode_change)

    def open_image(self):
        """Membuka dialog untuk memilih gambar"""
        file_path, _ = QFileDialog.getOpenFileName(self, "Open
Image", "", "Images (*.png *.jpg *.jpeg)")
        if file_path:
            self.image = cv2.imread(file_path)
            if self.image is None:
                print(f"⚠ Tidak dapat membaca gambar di
{file_path}")
                return

            self.processed_image =
self.detect_javanese_characters(self.image)
            self.display_image(self.processed_image)
            self.save_button.setEnabled(True)
```

```
            # Simpan path file untuk nanti
            self.current_file_path = file_path

    def detect_javanese_characters(self, image):
        """Deteksi aksara Jawa dan gambar bounding box"""
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        thresh = cv2.adaptiveThreshold(gray, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 15, 5)

        kernel = np.array([[0,1,0], [1,1,1], [0,1,0]],
dtype=np.uint8)
        dilated = cv2.dilate(thresh, kernel, iterations=1)

        # Deteksi kontur
        contours, _ = cv2.findContours(dilated,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        bounding_boxes = []
        for contour in contours:
            x, y, w, h = cv2.boundingRect(contour)

            if 8 < w < 60 and 10 < h < 80:
                aspect_ratio = w / h
                if 0.2 < aspect_ratio < 2.5:
                    bounding_boxes.append((x, y, w, h))

        merged_boxes = []
        bounding_boxes.sort()

        for i in range(len(bounding_boxes)):
            x, y, w, h = bounding_boxes[i]

            merged = False
            for j in range(len(merged_boxes)):
                mx, my, mw, mh = merged_boxes[j]

                if abs(mx - x) < 12 and abs(my - y) < 12:
                    merged_boxes[j] = (min(mx, x), min(my, y),
max(mx+mw, x+w) - min(mx, x), max(my+mh, y+h) - min(my, y))
                    merged = True
                    break

            if not merged:
                merged_boxes.append((x, y, w, h))
```

```
        # **Gambar bounding box setipis mungkin (ketebalan =
1)**
        for x, y, w, h in merged_boxes:
            cv2.rectangle(image, (x, y), (x + w, y + h), (0,
255, 0), 1)

        return image

    def display_image(self, img):
        """Menampilkan gambar di QLabel"""
        h, w, ch = img.shape
        bytes_per_line = ch * w
        q_image = QImage(img.data, w, h, bytes_per_line,
QImage.Format.Format_BGR888)

        pixmap = QPixmap.fromImage(q_image)
        self.image_label.setPixmap(pixmap)
        self.image_label.resize(pixmap.size())

    def save_image(self):
        """Menyimpan gambar dengan bounding box"""
        if self.processed_image is None:
            return

        # Tentukan path penyimpanan
        save_path = "E:/codefix/hasil_deteksi"
        if not os.path.exists(save_path):
            os.makedirs(save_path)

        # Ambil nama file tanpa path
        file_name = os.path.basename(self.current_file_path)
        output_path = os.path.join(save_path,
f"detected_{file_name}")

        # Simpan gambar
        cv2.imwrite(output_path, self.processed_image)
        print(f"☑ Gambar tersimpan di {output_path}")

    def handle_mode_change(self):
        """Menjalankan script berdasarkan pilihan di dropdown"""
        mode = self.mode_selector.currentText()
        script_path = None

        if mode == "Preprocessing":
```

```
            script_path =
os.path.join(os.path.dirname(__file__), "prepro2.py")
        elif mode == "Cropping":
            script_path =
os.path.join(os.path.dirname(__file__), "cropping.py")

        if script_path and os.path.exists(script_path):
            self.close()
            subprocess.Popen(["python", script_path])
        elif script_path:
            print(f"❌ {script_path} tidak ditemukan!")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = BoundingBoxApp()
    window.show()
    sys.exit(app.exec())
```

Segmentasi

```
import cv2
import numpy as np
import sys
import os
import subprocess
from PyQt6.QtWidgets import (
    QApplication, QWidget, QLabel, QPushButton, QFileDialog,
    QScrollArea, QVBoxLayout, QHBoxLayout, QComboBox
)
from PyQt6.QtCore import Qt
from PyQt6.QtGui import QPixmap, QImage

class CroppingApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Cropping from Bounding Box Image")
        self.setGeometry(100, 100, 1000, 600)

        # Variabel gambar
        self.image = None

        # Scroll Area untuk gambar besar
        self.scroll_area = QScrollArea(self)
```

```
        self.image_label = QLabel()
        self.image_label.setAlignment(Qt.AlignmentFlag.AlignCent
er)
        self.scroll_area.setWidget(self.image_label)
        self.scroll_area.setWidgetResizable(True)
        self.scroll_area.setFixedSize(600, 500)

        # Tombol utama
        self.open_button = QPushButton("Open Processed Image")
        self.crop_button = QPushButton("Crop & Save")
        self.crop_button.setEnabled(False)

        # Dropdown untuk memilih mode (Prepro2 / BoundingBox)
        self.mode_selector = QComboBox()
        self.mode_selector.addItem("Cropping")
        self.mode_selector.addItem("Preprocessing")
        self.mode_selector.addItem("BoundingBox")

        # Layout kiri (gambar)
        left_layout = QVBoxLayout()
        left_layout.addWidget(self.scroll_area,
alignment=Qt.AlignmentFlag.AlignCenter)

        # Layout kanan (tombol & dropdown)
        right_layout = QVBoxLayout()
        right_layout.addWidget(self.open_button)
        right_layout.addWidget(self.crop_button)
        right_layout.addWidget(self.mode_selector)
        right_layout.addStretch()

        # Layout utama (kiri-kanan)
        main_layout = QHBoxLayout()
        main_layout.addLayout(left_layout, 2)
        main_layout.addLayout(right_layout, 1)

        self.setLayout(main_layout)

        # Event handling
        self.open_button.clicked.connect(self.open_image)
        self.crop_button.clicked.connect(self.crop_and_save)
        self.mode_selector.currentIndexChanged.connect(self.hand
le_mode_change)

    def open_image(self):
```

```
        """Membuka dialog untuk memilih gambar hasil bounding
box"""
        file_path, _ = QFileDialog.getOpenFileName(self, "Open
Image", "", "Images (*.png *.jpg *.jpeg)")
        if file_path:
            self.image = cv2.imread(file_path)
            if self.image is None:
                print(f"⚠ Tidak dapat membaca gambar di
{file_path}")
                return

            self.display_image(self.image)
            self.crop_button.setEnabled(True)
            self.current_file_path = file_path

    def extract_bounding_boxes(self):
        """Mendeteksi bounding box dari gambar yang sudah ada
kotaknya"""
        if self.image is None:
            return []

        # Cari area hijau (bounding box)
        lower_green = np.array([0, 200, 0], dtype=np.uint8)
        upper_green = np.array([100, 255, 100], dtype=np.uint8)

        mask = cv2.inRange(self.image, lower_green, upper_green)
        contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

        bounding_boxes = []
        for contour in contours:
            x, y, w, h = cv2.boundingRect(contour)
            bounding_boxes.append((x, y, w, h))

        return bounding_boxes

    def crop_and_save(self):
        """Crop setiap bounding box dan simpan hasilnya dengan
penomoran berlanjut"""
        if self.image is None:
            return

        # Tentukan folder penyimpanan
        save_path = "E:/codefix/semhas_crop"
        if not os.path.exists(save_path):
```

```
            os.makedirs(save_path)

        # Cari nomor terakhir yang sudah ada
        existing_files = [f for f in os.listdir(save_path) if
f.startswith("crop_") and f.endswith(".png")]
        existing_numbers = []
        for fname in existing_files:
            try:
                num = int(fname.split("_")[1].split(".")[0])  #
ambil angka dari "crop_x.png"
                existing_numbers.append(num)
            except ValueError:
                continue

        start_idx = max(existing_numbers) + 1 if
existing_numbers else 1

        # Ambil bounding box dari gambar yang sudah ada kotaknya
        bounding_boxes = self.extract_bounding_boxes()

        # Simpan hasil cropping
        for idx, (x, y, w, h) in enumerate(bounding_boxes,
start=start_idx):
            cropped_image = self.image[y:y+h, x:x+w]
            output_path = os.path.join(save_path,
f"crop_{idx}.png")
            cv2.imwrite(output_path, cropped_image)

        print(f"☑ {len(bounding_boxes)} karakter berhasil di-
crop dan disimpan di {save_path}, mulai dari
crop_{start_idx}.png")

    def display_image(self, img):
        """Menampilkan gambar di QLabel"""
        h, w, ch = img.shape
        bytes_per_line = ch * w
        q_image = QImage(img.data, w, h, bytes_per_line,
QImage.Format.Format_BGR888)

        pixmap = QPixmap.fromImage(q_image)
        self.image_label.setPixmap(pixmap)
        self.image_label.resize(pixmap.size())

    def handle_mode_change(self):
        """Menjalankan script berdasarkan pilihan di dropdown"""
```

```
        mode = self.mode_selector.currentText()
        script_path = None

        if mode == "Preprocessing":
            script_path =
os.path.join(os.path.dirname(__file__), "prepro2.py")
        elif mode == "BoundingBox":
            script_path =
os.path.join(os.path.dirname(__file__), "boundingbox.py")

        if script_path and os.path.exists(script_path):
            self.close()
            subprocess.Popen(["python", script_path])
        elif script_path:
            print(f"❌ {script_path} tidak ditemukan!")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = CroppingApp()
    window.show()
    sys.exit(app.exec())
```

Clustering

```
import numpy as np
import cv2
from skimage.feature import hog
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
import shutil
import os

def extract_hog_features(image):
    """Ekstraksi fitur HOG dari gambar dengan ukuran minimal
64x64"""
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    resized_image = cv2.resize(gray_image, (64, 64))

    features, hog_image = hog(
        resized_image,
        pixels_per_cell=(8, 8),
        cells_per_block=(2, 2),
```

```
        orientations=9,
        visualize=True
    )

    print(f"Fitur untuk gambar: {len(features)}")
    return features


def visualize_dendrogram(features_array, max_samples=50):
    """
    Visualisasi proses penggabungan cluster (dendrogram) dengan
sampling otomatis.
    Sampling digunakan agar dendrogram tetap ringan untuk
dataset besar.
    """
    import random
    print("\n📌 Membuat dendrogram dengan sampling agar proses
tidak berat...")

    # Sampling jika data lebih banyak dari max_samples
    if len(features_array) > max_samples:
        indices = random.sample(range(len(features_array)),
max_samples)
        sampled_features = features_array[indices]
        print(f"🔁 Dataset berukuran besar, dendrogram dibuat
dari {max_samples} sampel acak.")
    else:
        sampled_features = features_array
        print(f"🔁 Dataset kecil, dendrogram dibuat dari
seluruh sampel.")

    linked = linkage(sampled_features, method="ward")

    plt.figure(figsize=(12, 6))
    dendrogram(
        linked,
        truncate_mode="level",
        p=25,
        leaf_rotation=90
    )
    plt.title("Dendrogram Agglomerative Hierarchical Clustering
(Sampling)")
    plt.xlabel("Sample Index / Cluster")
    plt.ylabel("Distance")
    plt.tight_layout()
```

```
    plt.show()



def perform_clustering(features_list):
    """Melakukan clustering menggunakan Agglomerative
Hierarchical Clustering"""
    clustering = AgglomerativeClustering(
        n_clusters=None,
        distance_threshold=12.0
    )
    clustering.fit(features_list)
    return clustering.labels_


def save_images_to_clusters(image_paths, cluster_labels):
    """Menyimpan gambar berdasarkan hasil clustering"""
    base_folder = "clustered_images12"

    if not os.path.exists(base_folder):
        os.makedirs(base_folder)

    for i, label in enumerate(cluster_labels):
        cluster_folder = os.path.join(base_folder,
f"Cluster_{label}")
        if not os.path.exists(cluster_folder):
            os.makedirs(cluster_folder)

        image_path = image_paths[i]
        image_name = os.path.basename(image_path)
        destination_path = os.path.join(cluster_folder,
image_name)
        shutil.copy(image_path, destination_path)

        print(f"Image {image_name} saved to Cluster {label}")


def process_and_cluster_images(cropped_image_folder):
    """Menangani ekstraksi fitur + dendrogram + clustering"""
    cropped_image_paths = [
        os.path.join(cropped_image_folder, f)
        for f in os.listdir(cropped_image_folder)
        if f.endswith('.png')
    ]
```

```
    features_list = []
    image_paths = []

    for image_path in cropped_image_paths:
        image = cv2.imread(image_path)
        if image is None:
            print(f"⚠ Gagal membaca gambar: {image_path}")
            continue

        features = extract_hog_features(image)
        features_list.append(features)
        image_paths.append(image_path)

    features_array = np.array(features_list)
    print(f"\nShape fitur array: {features_array.shape}")

    #  Visualisasi proses hierarkis sebelum clustering
    visualize_dendrogram(features_array)

    #  Clustering HOG fitur dengan AHC
    cluster_labels = perform_clustering(features_array)

    #  Simpan fitur & label untuk evaluasi di Bab 4
    np.save("E:/codefix/features12.npy", features_array)
    np.save("E:/codefix/labels12.npy", cluster_labels)
    print("\n☑ Fitur dan label berhasil disimpan (features.npy
& labels.npy)")

    #  Simpan gambar berdasarkan hasil cluster
    save_images_to_clusters(image_paths, cluster_labels)


# Tentukan folder tempat hasil cropping disimpan
cropped_image_folder = "E:/codefix/hasil_crop"
process_and_cluster_images(cropped_image_folder)
```

Evaluasi

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from sklearn.metrics import silhouette_score,
davies_bouldin_score
```

```
def visualize_tsne(features, labels, sample_size=2000):
    """Visualisasi hasil clustering menggunakan t-SNE."""
    # Ambil subset agar plotting tidak terlalu berat
    if len(features) > sample_size:
        idx = np.random.choice(len(features), sample_size,
replace=False)
        features = features[idx]
        labels = labels[idx]

    tsne = TSNE(n_components=2, perplexity=30, random_state=42)
    reduced = tsne.fit_transform(features)

    plt.figure(figsize=(10, 7))
    scatter = plt.scatter(reduced[:, 0], reduced[:, 1],
                          c=labels, cmap='tab20', s=10,
alpha=0.7)
    plt.title("Visualisasi Clustering (t-SNE)")
    plt.xlabel("t-SNE Dim 1")
    plt.ylabel("t-SNE Dim 2")
    plt.colorbar(scatter, label="Cluster ID")
    plt.show()


def evaluate_clustering(features_path, labels_path):
    """Evaluasi clustering hanya dengan Silhouette Score &
Davies-Bouldin Index."""
    features = np.load(features_path)
    labels = np.load(labels_path)

    unique_labels = np.unique(labels)
    n_clusters = len(unique_labels)
    n_samples = len(features)

    print("\n=== INFORMASI DASAR CLUSTER ===")
    print(f"Jumlah sampel  : {n_samples}")
    print(f"Jumlah cluster : {n_clusters}")

    if n_clusters < 2:
        print("⚠ Tidak cukup cluster untuk evaluasi (minimal 2
cluster).")
        return

    # ---- Evaluasi metrik ----
    sil_score = silhouette_score(features, labels)
```

```
    dbi_score = davies_bouldin_score(features, labels)

    print("\n=== EVALUASI CLUSTERING ===")
    print(f"Silhouette Score         : {sil_score:.6f}")
    print(f"Davies-Bouldin Index     : {dbi_score:.6f}")

    print("\n=== INTERPRETASI ===")
    print("✔ Silhouette tinggi → cluster makin terpisah.")
    print("✔ DBI rendah → cluster makin kompak dan
terstruktur.")

    # ---- Visualisasi ----
    print("\n📊 Membuat visualisasi hasil clustering (t-
SNE)...")
    visualize_tsne(features, labels)


# Jalankan evaluasi
features_path = "E:/codefix/features13.npy"
labels_path = "E:/codefix/labels13.npy"
evaluate_clustering(features_path, labels_path)
```