LAPORAN TUGAS BESAR III

IF2211 STRATEGI ALGORITMA

"Pemanfaatan Pattern Matching dalam Membangun Sistem Deteksi Individu Berbasis Biometrik Melalui Citra Sidik Jari"

Dosen:

Ir. Rila Mandala, M. Eng, Ph. D.

Monterico Adrian, S.T., M.T.

Kelompok 54:

13522125 Satriadhikara Panji Yudhistira

13522140 Yasmin Farisah Salma

13522145 Farrel Natha Saskoro

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2023/2024

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa penulis ucapkan atas keberhasilan dalam menyelesaikan Tugas Besar II IF2123 Aljabar Linear dan Geometri ini.

Puji syukur kepada Tuhan Yang Maha Esa penulis ucapkan atas keberhasilan dalam menyelesaikan Tugas Besar III IF2211 Strategi Algoritma ini. Laporan ini merangkum seluruh fase pengembangan sistem deteksi individu berbasis biometrik melalui citra sidik jari, mulai dari konsepsi ide hingga realisasi teknis.

Lebih dari sekadar tugas akademis, proyek ini adalah sebuah perjalanan pembelajaran tentang bagaimana konsep pattern matching dan algoritma pencocokan string dapat diaplikasikan dalam memecahkan masalah dunia nyata—dalam hal ini, menciptakan sistem identifikasi sidik jari yang canggih dan user-friendly. Kami berkomitmen untuk menyajikan sebuah platform yang tidak hanya akurat dalam mengidentifikasi individu berdasarkan sidik jari, tetapi juga menyediakan pengalaman pengguna yang intuitif dan memuaskan. Pengembangan sistem ini juga merupakan hasil atas kolaborasi dan kerja tim kami yang di dalamnya terjalin sinergi antara pengetahuan teoretis dan keterampilan teknis.

Dengan dukungan dari dosen, asisten kelompok, input dari rekan mahasiswa, dan berbagai sumber daya yang tersedia, kami menghadirkan sebuah sistem deteksi individu berbasis biometrik yang kami harap akan bekerja dengan baik dan efisien. Akhir kata, kami ucapkan terima kasih kepada semua yang telah berpartisipasi dan mendukung. Kami berharap agar tugas besar ini dapat memenuhi mata kuliah IF2211 Strategi Algoritma dan bermanfaat untuk pengembangan yang lebih maju.

Sumedang, 6 Juni 2024,

The Tortured Informatics Department.

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

1.1. Deskripsi Masalah

Di era digital ini, keamanan data dan akses menjadi semakin penting. Perkembangan teknologi membuka peluang untuk berbagai metode identifikasi yang canggih dan praktis. Beberapa metode umum yang sering digunakan seperti kata sandi atau pin, namun memiliki kelemahan seperti mudah terlupakan atau dicuri. Oleh karena itu, biometrik menjadi alternatif metode akses keamanan yang semakin populer. Salah satu teknologi biometrik yang banyak digunakan adalah identifikasi sidik jari. Sidik jari setiap orang memiliki pola yang unik dan tidak dapat ditiru, sehingga cocok untuk digunakan sebagai identitas individu.

Pada tugas ini, kami telah mengimplementasikan sebuah program yang dapat melakukan identifikasi biometrik berbasis sidik jari. Proses implementasi dilakukan dengan menggunakan algoritma Boyer-Moore dan Knuth-Morris-Pratt, sesuai dengan yang diajarkan pada materi dan salindia kuliah.

Secara sekilas, penggunaan algoritma *pattern matching* dalam mencocokkan sidik jari terdiri atas tiga tahapan utama dengan skema sebagai berikut.

Secara umum, penggunaan algoritma *pattern matching* dalam mencocokkan sidik jari terdiri atas tiga tahapan utama. Tahap pertama adalah pengambilan dan konversi citra sidik jari, di mana gambar sidik jari dalam format bitmap (.bmp) dan dikonversi menjadi data biner. Data biner tersebut

kemudian dikelompokkan per 8 bit sehingga membentuk karakter ASCII 8-bit yang merepresentasikan sebuah sidik jari.

Tahap kedua adalah proses *pattern matching*, di mana karakter ASCII 8-bit tersebut dicocokkan dengan string data yang ada di basis data menggunakan algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Jika tidak ada sidik jari pada basis data yang *match*, sistem akan menggunakan sidik jari paling mirip dengan kesamaan di atas nilai tertentu (*threshold*). Tahap ketiga adalah penanganan data korup pada atribut nama di basis data, di mana kami menggunakan Regular Expression (Regex) untuk mengembalikan pola karakter alay ke bentuk alfabetik yang bersesuaian. Setelah itu, sidik jari yang paling mirip akan dicocokkan dengan daftar sidik jari di basis data dan menampilkan biodata yang sesuai jika ditemukan kecocokan di atas batas persentase tertentu.

Selain itu, kami juga mengembangkan antarmuka pengguna yang *user-friendly* untuk aplikasi desktop menggunakan C#. Antarmuka ini menyediakan fitur seperti memasukkan citra sidik jari, memilih algoritma pencarian (KMP atau BM), dan menampilkan hasil pencarian. Langkah-langkah ini diharapkan dapat menghasilkan sistem identifikasi individu berbasis biometrik yang aman, handal, dan mudah digunakan.

BAB II
LANDASAN TEORI

2.1 Knuth Morris Pratt Algorithm

Algoritma Knuth-Morris-Pratt (KMP) adalah salah satu algoritma pencocokan pola yang digunakan untuk menemukan lokasi kemunculan sebuah pola (pattern) dalam sebuah teks. Algoritma ini dikembangkan oleh Donald Knuth, Vaughan Pratt, dan James H. Morris pada tahun 1977. KMP memperbaiki kelemahan algoritma brute force dengan cara yang lebih efisien dalam menggeser pola ketika terjadi ketidakcocokan karakter, sehingga menghindari perbandingan yang berulang-ulang.

Algoritma KMP melakukan pencocokan pola dalam teks dengan urutan dari kiri ke kanan, mirip dengan algoritma brute force. Namun, perbedaannya terletak pada cara algoritma ini menggeser pola ketika ditemukan ketidakcocokan karakter. Algoritma KMP menggunakan informasi dari pola itu sendiri untuk menentukan sejauh mana pola dapat digeser tanpa melakukan perbandingan yang tidak perlu.

Misalkan kita memiliki teks T: "abacaabacabacababa" dan pola P: "acabaca". Algoritma KMP akan memproses pola terlebih dahulu untuk menemukan fungsi pinggiran (border function) yang dapat digunakan untuk menggeser pola dengan lebih efisien ketika terjadi ketidakcocokan.

Fungsi pinggiran atau border function digunakan untuk menemukan ukuran prefiks terbesar dari pola yang juga merupakan sufiks dari pola itu sendiri. Fungsi ini dihitung terlebih dahulu dalam tahap preproses pola. Fungsi ini diwakili oleh array b[], di mana b[k] adalah ukuran prefiks terbesar dari P[0..k] yang juga merupakan sufiks dari P[1..k].

Selama proses pencocokan, algoritma KMP menggunakan fungsi pinggiran untuk menggeser pola ke kanan dengan cara yang lebih cerdas ketika ditemukan ketidakcocokan antara karakter dalam teks dan pola. Jika terjadi ketidakcocokan pada posisi P[j] dalam pola, algoritma akan menggeser pola sejauh j - b[j-1].

Algoritma KMP memiliki kompleksitas waktu O(m + n), di mana m adalah panjang pola dan n adalah panjang teks. Kompleksitas ini jauh lebih efisien dibandingkan dengan algoritma brute force yang memiliki kompleksitas O(m * n) pada kasus terburuk.

Keuntungan utama dari algoritma KMP adalah efisiensinya dalam pencocokan pola, karena tidak pernah perlu bergerak mundur dalam teks. Hal ini membuat algoritma KMP sangat cocok untuk memproses berkas berukuran besar yang dibaca dari perangkat eksternal atau melalui jaringan. Namun, kelemahan dari algoritma KMP adalah kinerjanya yang kurang optimal saat ukuran alfabet besar, karena kemungkinan ketidakcocokan karakter lebih banyak terjadi pada awal pola.

2.2 Boyer Moore Algorithm

Algoritma Boyer-Moore adalah salah satu algoritma pencocokan pola yang efisien dan sering digunakan dalam pencarian teks. Algoritma ini dikembangkan oleh Robert S. Boyer dan J Strother Moore pada tahun 1977. Boyer-Moore dikenal sebagai salah satu algoritma pencarian string tercepat dalam praktik, terutama ketika digunakan untuk mencari pola dalam teks yang panjang.

Algoritma Boyer-Moore didasarkan pada dua teknik utama: teknik cermin (looking-glass) dan teknik lompatan karakter (character-jump). Algoritma ini melakukan pencarian pola dari kanan ke kiri, dimulai dari akhir pola. Teknik *looking-glass* mencari pola dalam teks dengan bergerak mundur

melalui pola, dimulai dari ujungnya. Jika terjadi ketidakcocokan antara karakter dalam teks dan pola, algoritma menggunakan informasi tentang posisi ketidakcocokan untuk menentukan seberapa jauh pola harus digeser. Ketika terjadi ketidakcocokan pada karakter dalam teks dan pola, algoritma Boyer-Moore menggunakan fungsi last occurrence untuk menentukan posisi terakhir dari karakter yang tidak cocok dalam pola. Pola kemudian digeser ke kanan sehingga karakter yang tidak cocok pada teks sejajar dengan kemunculan terakhirnya dalam pola.

Ada tiga kemungkinan kasus pergeseran yang dihadapi algoritma Boyer-Moore:

1. Jika pola mengandung karakter yang tidak cocok di suatu tempat, geser pola ke kanan untuk menyelaraskan kemunculan terakhir karakter tersebut dengan posisi karakter dalam teks.
2. Jika pola mengandung karakter yang tidak cocok tetapi tidak ada kemunculan lebih lanjut, geser pola ke kanan satu karakter.
3. Jika karakter yang tidak cocok tidak ada dalam pola, geser pola ke kanan untuk menyelaraskan awal pola dengan posisi berikutnya dalam teks.

Fungsi last occurrence memetakan semua karakter dalam alfabet ke indeks terbesar dalam pola di mana karakter tersebut muncul. Fungsi ini digunakan untuk menentukan seberapa jauh pola harus digeser ketika terjadi ketidakcocokan.

Kompleksitas waktu terburuk algoritma Boyer-Moore adalah O(nm + A), di mana n adalah panjang teks, m adalah panjang pola, dan A adalah ukuran alfabet. Namun, dalam praktik, algoritma ini sangat cepat untuk pencarian teks bahasa Inggris dan biasanya memiliki kinerja lebih baik daripada algoritma brute force.

Keuntungan utama dari algoritma Boyer-Moore adalah kemampuannya untuk menggeser pola dengan jarak yang lebih jauh, mengurangi jumlah perbandingan karakter secara signifikan, dan tidak perlu bergerak mundur dalam teks. Namun, algoritma ini tidak begitu efisien untuk alfabet kecil, seperti biner, karena lebih banyak kemungkinan ketidakcocokan terjadi.

2.3 Regular Expression

Regular Expression (Regex) adalah sekumpulan karakter yang membentuk pola pencarian. Regex digunakan untuk melakukan pencocokan string, penggantian string, atau pemisahan string berdasarkan pola tertentu. Regex sangat berguna dalam pengolahan teks dan sering digunakan dalam pemrograman untuk memvalidasi input, mencari teks dalam dokumen, dan berbagai operasi string lainnya.

Regex terdiri dari berbagai notasi yang memungkinkan pencarian dan manipulasi teks secara efisien. Misalnya, karakter titik (.) dalam Regex mewakili sembarang karakter tunggal, dan tanda bintang (*) menunjukkan nol atau lebih kemunculan dari karakter sebelumnya.

1. Brackets []: Disjunction atau pemilihan karakter di dalamnya.
2. Caret ^: Menandakan negasi jika digunakan di dalam brackets, atau awal string jika digunakan di luar brackets.
3. Tanda tanya ?: Menandakan bahwa karakter sebelumnya bisa ada atau tidak.
4. Titik .: Menandakan sembarang karakter tunggal.
5. **Asterisk ***: Menandakan nol atau lebih kemunculan karakter sebelumnya.
6. Plus +: Menandakan satu atau lebih kemunculan karakter sebelumnya.

7. Curly Braces {}: Menandakan jumlah kemunculan karakter sebelumnya dalam rentang tertentu.

Regex adalah alat yang sangat kuat untuk pencocokan dan manipulasi string. Dengan menggunakan berbagai notasi dan pola, Regex dapat digunakan untuk berbagai tujuan mulai dari validasi input hingga pengolahan teks yang kompleks.

2.4 Pengukuran Persentase Kemiripan

Pengukuran persentase kemiripan antara sidik jari input (pattern) dan sidik jari *database* (text) yang diberikan dalam program ini dilakukan dengan menggunakan algoritma pencocokan string Boyer-Moore atau Knuth-Morris-Pratt. Pertama, program menerima dua input dari pengguna, yaitu teks lengkap yang merupakan string panjang dan pola yang merupakan string yang lebih pendek. Setelah input diterima, program akan memilih algoritma pencocokan string yang diinginkan berdasarkan pilihan pengguna. Algoritma yang dipilih kemudian akan mencari semua kemunculan pola dalam teks lengkap. Setiap kali pola ditemukan dalam teks, program akan mencatat jumlah karakter yang cocok dari pola tersebut.

Selanjutnya, program menghitung total karakter yang cocok dengan menjumlahkan semua karakter yang cocok dari setiap kemunculan pola dalam teks. Persentase kemiripan dihitung dengan membagi total karakter yang cocok dengan jumlah total karakter dalam teks lengkap. Hasil pembagian ini kemudian dikalikan dengan 100 untuk mendapatkan hasil dalam bentuk persentase.

2.5 Penjelasan Aplikasi Desktop

Avalonia adalah sebuah framework UI open-source yang digunakan untuk membangun aplikasi desktop lintas platform dengan .NET. Framework ini mendukung berbagai sistem operasi seperti Windows, macOS, dan Linux, memungkinkan pengembang untuk menulis aplikasi dengan satu kode dasar yang dapat dijalankan di berbagai sistem operasi tanpa modifikasi signifikan. Avalonia menggunakan XAML (eXtensible Application Markup Language) untuk mendefinisikan antarmuka pengguna, mirip dengan WPF (Windows Presentation Foundation), sehingga memudahkan pengembang yang sudah familiar dengan teknologi Microsoft. Selain itu, Avalonia mendukung pola desain Model-View-ViewModel (MVVM) yang memisahkan logika bisnis dari antarmuka pengguna, membuat kode lebih terstruktur dan mudah diuji.

Pengembangan aplikasi dengan Avalonia melibatkan beberapa langkah dasar seperti setup proyek, definisi UI dengan XAML, implementasi logika bisnis dengan C#, binding data, serta styling dan theming. Misalnya, pengembang dapat membuat proyek baru menggunakan template Avalonia yang disediakan oleh .NET CLI atau Visual Studio, kemudian menggunakan XAML untuk mendefinisikan tata letak dan elemen UI seperti window utama, panel, tombol, teks, dan gambar. Logika bisnis dan interaksi pengguna diimplementasikan dengan menggunakan bahasa pemrograman C#, termasuk penanganan event, manipulasi data, dan komunikasi dengan layanan backend. Binding data digunakan untuk menghubungkan data model dengan elemen UI, memudahkan sinkronisasi data antara logika aplikasi dan tampilan. Pengembang juga dapat menyesuaikan tampilan aplikasi dengan mendefinisikan gaya (styles) dan tema (themes) untuk kontrol UI, sehingga aplikasi memiliki tampilan yang konsisten dan menarik.

BAB III
ANALISIS PEMECAHAN MASALAH

3.1 Langkah-Langkah Pemecahan Masalah

Langkah-langkah pemecahan masalah dalam tugas besar ini dimulai dengan identifikasi kebutuhan dan spesifikasi sistem. Langkah awal ini melibatkan pemahaman latar belakang dan tujuan tugas besar yang mengharuskan implementasi sistem identifikasi individu berbasis biometrik melalui citra sidik jari. Kami meninjau spesifikasi yang diberikan, termasuk penggunaan algoritma Boyer-Moore dan Knuth-Morris-Pratt untuk pencocokan pola, serta penggunaan basis data SQL untuk menyimpan dan mencocokkan data sidik jari dengan biodata individu. Selanjutnya, kami melakukan perencanaan dan desain sistem yang mencakup desain struktur basis data untuk menyimpan data sidik jari dan biodata individu sesuai skema yang disediakan.

Implementasi algoritma pencocokan pola menjadi tahap berikutnya, di mana kami mengimplementasikan algoritma Boyer-Moore dan Knuth-Morris-Pratt dalam bahasa C# sesuai dengan materi kuliah. Kami menambahkan fungsi untuk menghitung persentase kemiripan antara citra sidik jari yang dimasukkan dengan data sidik jari yang ada di basis data. Setelah itu, kami melakukan konversi citra sidik jari yang diunggah pengguna menjadi data biner, kemudian mengelompokkan setiap baris kode biner per 8 bit untuk membentuk karakter ASCII.

Kami juga menangani kemungkinan adanya data korup dengan menggunakan Regular Expression (Regex) untuk menangani variasi bahasa alay dalam data nama. Setelah konversi dari bahasa alay ke bentuk alfabetik yang sesuai, pencocokan nama dilakukan dengan algoritma KMP atau BM. Langkah selanjutnya adalah mengintegrasikan program dengan basis data

SQL yang telah dirancang, mengimplementasikan fitur untuk menyimpan citra sidik jari baru ke dalam basis data, dan mencocokkannya dengan citra yang sudah ada.

3.2 Proses Penyelesaian Solusi dengan Knuth Morris Pratt Algorithm

Proses penyelesaian solusi dengan Algoritma Knuth-Morris-Pratt (KMP) dimulai dengan inisialisasi dan preprocessing pola. Pada tahap ini, dilakukan inisialisasi pola dan pembangunan tabel border yang akan digunakan untuk mempercepat proses pencocokan pola. Tabel border menyimpan informasi tentang panjang prefix terbesar dari pola yang juga merupakan suffix. Proses ini dilakukan dalam metode ComputeBorder. Tabel border diinisialisasi dengan panjang yang sama dengan pola, dan semua nilainya diatur menjadi 0. Selanjutnya, iterasi dilakukan melalui karakter-karakter dalam pola untuk mengisi tabel border. Jika karakter pada posisi saat ini cocok dengan karakter pada posisi sebelumnya dalam pola, nilai pada tabel border diperbarui dengan panjang prefix yang cocok. Jika tidak cocok, indeks j dikembalikan ke nilai border sebelumnya, dan proses pencocokan dilanjutkan.

Setelah tabel border selesai dibuat, langkah berikutnya adalah mencocokkan pola dalam teks menggunakan tabel border. Proses pencocokan ini dilakukan dalam metode Search. Algoritma KMP membandingkan karakter-karakter pola dengan karakter-karakter teks dari kiri ke kanan. Jika terjadi ketidakcocokan, algoritma menggunakan informasi dari tabel border untuk menggeser pola sehingga perbandingan dapat dilanjutkan tanpa harus mengulangi perbandingan karakter yang sudah dilakukan sebelumnya. Dalam proses ini, dua indeks diinisialisasi, satu untuk teks (i) dan satu untuk pola (j). Jika karakter dalam teks cocok dengan karakter dalam pola, kedua indeks

meningkat. Jika terjadi ketidakcocokan, indeks j diatur ke nilai border sebelumnya, dan proses pencocokan dilanjutkan. Jika seluruh karakter dalam pola cocok dengan bagian dari teks, indeks posisi awal pencocokan dikembalikan. Jika tidak ditemukan kecocokan, metode mengembalikan -1.

Selain mencocokkan pola, program juga menghitung persentase kemiripan antara teks dan pola. Proses ini dilakukan dalam metode CalculateSimilarity. Metode ini mencari semua kemunculan pola dalam teks dan menghitung total karakter yang cocok. Persentase kemiripan dihitung dengan membagi total karakter yang cocok dengan jumlah total karakter dalam teks, kemudian dikalikan dengan 100.

3.3 Proses Penyelesaian Solusi dengan Boyer Moore Algorithm

Proses penyelesaian solusi dengan Algoritma Boyer-Moore (BM) dimulai dengan inisialisasi dan preprocessing pola. Pada tahap ini, dilakukan inisialisasi pola dan pembangunan tabel last occurrence yang akan digunakan untuk mempercepat proses pencocokan pola. Tabel last occurrence menyimpan informasi tentang posisi terakhir setiap karakter dalam pola. Proses ini dilakukan dalam metode BuildLast, di mana tabel last diinisialisasi dengan panjang 128, sesuai dengan jumlah karakter dalam set ASCII, dan semua nilainya diatur menjadi -1. Selanjutnya, iterasi dilakukan melalui karakter-karakter dalam pola untuk mengisi tabel last dengan posisi terakhir kemunculan setiap karakter.

Setelah tabel last occurrence selesai dibuat, langkah berikutnya adalah mencocokkan pola dalam teks menggunakan tabel last. Proses pencocokan ini dilakukan dalam metode Search. Algoritma Boyer-Moore membandingkan karakter-karakter pola dengan karakter-karakter teks dari kanan ke kiri. Jika terjadi ketidakcocokan, algoritma menggunakan informasi dari tabel last untuk menggeser pola sehingga perbandingan dapat dilanjutkan dengan lebih cepat.

Dalam proses ini, dua indeks diinisialisasi, satu untuk teks (i) yang dimulai dari akhir pola, dan satu untuk pola (j) yang juga dimulai dari akhir pola. Jika karakter dalam teks cocok dengan karakter dalam pola, kedua indeks menurun. Jika terjadi ketidakcocokan, indeks i digeser ke kanan berdasarkan nilai dalam tabel last, dan indeks j diatur kembali ke posisi akhir pola. Jika seluruh karakter dalam pola cocok dengan bagian dari teks, indeks posisi awal pencocokan dikembalikan. Jika tidak ditemukan kecocokan, metode mengembalikan -1.

Selain mencocokkan pola, program juga menghitung persentase kemiripan antara teks dan pola. Proses ini dilakukan dalam metode CalculateSimilarity. Metode ini mencari semua kemunculan pola dalam teks dan menghitung total karakter yang cocok. Persentase kemiripan dihitung dengan membagi total karakter yang cocok dengan jumlah total karakter dalam teks, kemudian dikalikan dengan 100.

3.4 Fitur Fungsional dan Arsitektur Aplikasi Desktop

Adapun fitur fungsional dari aplikasi fingertape antara lain:

1. **Navigasi Antarmuka Pengguna**:
    - **Home Panel**: Menampilkan judul aplikasi dan gambar.
    - **Fingerprint Matching Panel**: Menyediakan fitur untuk memilih gambar sidik jari, memilih algoritma pencocokan (KMP atau BM), dan menampilkan hasil pencocokan serta waktu eksekusi.
    - **About Us Panel**: Menampilkan informasi tentang pengembang aplikasi.
2. **Interaksi Pengguna**:
    - **Button Home**: Mengarahkan pengguna ke panel beranda.

    - **Button Fingerprint**: Mengarahkan pengguna ke panel pencocokan sidik jari.
    - **Button Person**: Mengarahkan pengguna ke panel about us.
    - **Button Power**: Menutup aplikasi.
    - **Button Select Image**: Membuka dialog untuk memilih gambar sidik jari yang akan dicocokkan.
    - **Button Submit**: Mengeksekusi algoritma pencocokan dan menampilkan hasilnya.
3. **Algoritma Pencocokan Sidik Jari**:
    - **Knuth-Morris-Pratt (KMP)**: Digunakan untuk mencari kecocokan string dalam teks.
    - **Boyer-Moore (BM)**: Alternatif algoritma pencocokan string dengan performa berbeda.

Adapun arsitektur dari aplikasi fingertape antara lain:

1. **Struktur XAML**:
    - **<Window>**: Root elemen untuk jendela aplikasi dengan properti seperti ukuran, judul, dan dekorasi.
    - **<Panel>**: Kontainer utama dengan background transparan.
    - **<Border>**: Membungkus dan menata tampilan dalam grid.
    - **<Grid>**: Menyusun elemen UI dalam kolom dan baris.
2. **StackPanel dan Grid**:
    - **Navigasi Panel**: Terletak di bagian kiri untuk navigasi antar panel.
    - **Content Panel**: Di sebelah kanan, menampilkan konten utama berdasarkan navigasi yang dipilih.
3. **Pengendalian UI**:

- **Button Event Handlers**: Mengikat event klik ke fungsi yang sesuai di backend untuk mengubah tampilan panel atau memproses pencocokan gambar.

4. **Logika Backend**:
    - **MainWindow Class**: Mengelola logika aplikasi seperti inisialisasi komponen, event handler untuk klik tombol, dan algoritma pencocokan.
    - **Event Handlers**: Mengatur logika navigasi dan interaksi pengguna.
    - **Algoritma Pencocokan**: Menggunakan algoritma KMP atau BM untuk membandingkan representasi ASCII dari gambar sidik jari yang dipilih dengan gambar yang tersimpan.
5. **Manipulasi Gambar**:
    - **OpenFileDialog**: Membuka dialog untuk memilih file gambar.
    - **Bitmap Processing**: Mengkonversi gambar menjadi representasi biner dan ASCII untuk diproses oleh algoritma pencocokan.
6. **Desain Responsif**:
    - **Visibility Management**: Mengelola visibilitas panel sesuai dengan navigasi pengguna.

3.5 Ilustrasi Kasus dan Penyelesaian

- Seorang pengguna memilih gambar fingerprint
- Pengguna memilih algortima apa yang akan dipakai
- Sistem mengubah
- Perangkat lunak membandingkan hasil GLCM dan menampilkan berapa persen kesamaan kedua gambar dan time execution-nya

Penyelesaian:

Implementasi GLCM untuk ekstraksi parameter warna dan tekstur dan perbandingan hasil GLCM untuk menampilkan perbedaan antara gambar-gambar yang diunggah yang akan dijelaskan di bab empat.

BAB IV
IMPLEMENTASI DAN UJI COBA

4.1 Struktur Data, Fungsi, dan Prosedur

Spesifikasi teknis program (struktur data, fungsi, dan prosedur yang dibangun).

4.2 Penjelasan Tata Cara Penggunaan Program

1. Click tombol fingerprint untuk berpindah ke page fingerprint matching
2. Click tombol select image untuk memilih fingerprint yang akan dicari
3. Click tombol algorithm untuk memilih algoritma yang akan dioakai
4. Click tombol submit
5. Fingerprint ynag match akan muncul beserta dengan biodatanya, persen kecocokannya, dan waktu eksekusinya.

4.3 Hasil Pengujian

Hasil pengujian (screenshot antarmuka dan skenario yang memperlihatkan berbagai kasus yang mencakup seluruh fitur pada aplikasi Anda).

4.4 Analisis Hasil Pengujian

Dari hasil pengujian didapat hasil bahwa algoritma KMP lebih konsisten dalam performa dengan waktu terburuk yang lebih baik, menjadikannya pilihan yang lebih baik untuk teks atau pola yang panjang dan ketika pola sering muncul dalam teks, sedangkan BM cenderung lebih cepat dalam praktik dengan waktu rata-rata yang sangat baik, membuatnya ideal untuk pencarian dalam teks panjang dengan pola pendek dan ketika pola jarang muncul.

BAB V
PENUTUP

5.1 Kesimpulan

Pada tugas besar ini, kami telah berhasil mengembangkan dan menguji sistem deteksi individu berbasis biometrik melalui citra sidik jari. Sistem ini menggunakan algoritma Boyer-Moore dan Knuth-Morris-Pratt untuk pencocokan pola, yang memungkinkan pencarian sidik jari dengan efisien dan akurat. Implementasi ini membuktikan bahwa teknologi pattern matching dapat diaplikasikan dalam sistem identifikasi biometrik yang aman dan user-friendly. Penggunaan basis data SQL untuk menyimpan data sidik jari dan biodata individu juga memastikan integritas dan kemudahan akses data. Hasil pengujian menunjukkan bahwa sistem ini mampu mengenali sidik jari dengan tingkat kemiripan yang tinggi dan memberikan hasil yang konsisten.

5.2 Saran

Untuk pengembangan lebih lanjut, kami menyarankan peningkatan pada kapasitas dan keamanan basis data. Implementasi enkripsi data dapat meningkatkan privasi dan keamanan informasi yang disimpan. Selain itu, peningkatan pada antarmuka pengguna dengan menambahkan fitur seperti pengenalan sidik jari real-time dan integrasi dengan sistem lain dapat meningkatkan fungsionalitas dan pengalaman pengguna. Penggunaan teknologi pembelajaran mesin juga bisa dijajaki untuk meningkatkan akurasi pencocokan pola dan adaptasi terhadap variasi sidik jari yang lebih luas.

5.3 Komentar atau Tanggapan

Kami sangat menghargai setiap masukan yang konstruktif dari dosen, asisten, dan rekan mahasiswa yang dapat membantu dalam penyempurnaan sistem ini. Umpan balik yang objektif dan relevan sangat penting untuk mengidentifikasi kelemahan dan peluang perbaikan dalam sistem yang telah kami kembangkan. Kami berharap sistem ini dapat terus berkembang dan memberikan kontribusi nyata dalam bidang identifikasi biometrik.

5.4 Refleksi terhadap Tugas Besar

Melalui pengerjaan tugas besar ini, kami telah mempelajari banyak hal, mulai dari konsep teoritis hingga aplikasi praktis dalam pengembangan sistem yang kompleks. Tantangan yang dihadapi selama proses pengembangan telah mengasah kemampuan kami dalam bekerja sama, manajemen waktu, dan penyelesaian masalah. Kami juga belajar untuk lebih tangguh dan adaptif dalam menghadapi kendala teknis dan non-teknis. Pengalaman ini sangat berharga dan memberikan wawasan baru tentang pentingnya sinergi antara teori dan praktik dalam menciptakan solusi teknologi yang efektif.

5.5 Ruang Perbaikan atau Pengembangan

Meskipun sistem yang kami kembangkan sudah cukup baik, kami menyadari bahwa masih ada ruang untuk perbaikan dan pengembangan lebih lanjut. Optimalisasi algoritma pencocokan pola dan peningkatan efisiensi waktu eksekusi merupakan aspek yang perlu diperhatikan. Selain itu, desain antarmuka pengguna dapat ditingkatkan untuk memberikan pengalaman yang lebih intuitif dan responsif. Kami juga merencanakan untuk melakukan lebih banyak pengujian pengguna untuk mendapatkan umpan balik yang lebih

komprehensif, sehingga sistem ini dapat terus diperbaiki dan disempurnakan sesuai dengan kebutuhan pengguna.

DAFTAR PUSTAKA

Jun Yue, Z. L. (2011, August). *Content-based image retrieval using color and texture fused features*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S0895717710005352

MakersOfTailwindCSS. (n.d.). *Beautiful hand-crafted SVG icons, by the makers of Tailwind CSS.* Retrieved from heroicons: https://heroicons.com/

Munir, R. (2023). *Aljabar Linier dan Geometri.* Retrieved from Homepage Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi.munir/

RGB to HSV Color Conversion Algorithm. (2021). Retrieved from StackExchange: https://math.stackexchange.com/questions/556341/rgb-to-hsv-color-conversion-algorithm

Siagian, P., & Fernando, E. (2012). *KLASIFIKASI CITRA CONTENT-BASED IMAGE RETRIEVAL DENGAN METODE SHAPE BASE*

THRESHOLD. Retrieved from SNTIKI Seminar Nasional Teknologi Informasi Komunikasi dan Industri: https://ejournal.uin-suska.ac.id/index.php/SNTIKI/article/view/2846

Smyth, P. (2018, April 2). *Creating Web APIs with Python and Flask*. Retrieved from Programming Historian: https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask

Warongan, T. S. (2018). *Penerapan Metode Content-Based Image.* Retrieved from E-Journal UNSRAT: https://ejournal.unsrat.ac.id/index.php/informatika/article/download/28070/27542

Yunus, M. (2020, July 16). *Feature Extraction : Gray Level Co-occurrence Matrix (GLCM)*. Retrieved from Medium: https://yunusmuhammad007.medium.com/feature-extraction-gray-level-co-occurrence-matrix-glcm-10c45b6d46a1

LAMPIRAN

6.1 GitHub Repository

https://github.com/caernations/Tubes3_TheTorturedInformaticsDepartment