

**Tugas Besar 3 IF2211 Strategi Algoritma
Semester II tahun 2023/2024**

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



Disusun Oleh:

Ariel Herfrison 13522002

Eduardus Alvito Kristiadi 13522004

M Athaullah Daffa Kusuma M 13522044

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024**

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1	
DESKRIPSI TUGAS.....	4
BAB 2	
LANDASAN TEORI.....	5
A. Algoritma Knuth-Morris-Pratt (KMP).....	5
B. Algoritma Boyer-Moore (BM).....	5
C. Regex.....	5
D. Pengukuran Persentase Kemiripan.....	5
E. Penjelasan Aplikasi Desktop yang Dibangun.....	6
BAB 3	
ANALISIS PEMECAHAN MASALAH.....	7
A. Langkah-Langkah Pemecahan Masalah.....	7
B. Proses Penyelesaian Solusi dengan algoritma KMP dan BM.....	7
B.1. KMP.....	7
B.2. BM.....	8
C. Fitur Fungsional dan Arsitektur Aplikasi.....	9
D. Contoh Ilustrasi Kasus.....	10
D.1. Kasus 1.....	10
D.2. Kasus 2.....	10
D.3. Kasus 3.....	11
BAB 4	
IMPLEMENTASI DAN PENGUJIAN.....	12
A. Spesifikasi Teknis Program (struktur data, fungsi, dan prosedur yang dibangun).....	12
A.1. Kelas MainForm.....	12
A.2. Kelas Converter.....	18
A.3. Kelas Boyer Moore.....	22
A.4. Kelas KMPString.....	23
A.5. Kelas DatabaseManager.....	24
B. Tata Cara Penggunaan Program.....	25
C. Hasil Pengujian.....	27
C.1. Kasus Uji 1.....	27
C.2. Kasus Uji 2.....	27
C.3. Kasus Uji 3.....	28
C.4. Kasus Uji 4.....	28
C.5. Kasus Uji 5.....	29
C.6. Kasus Uji 6.....	29
C.7. Kasus Uji 7.....	30
C.8. Kasus Uji 8.....	30

D. Analisis Hasil Pengujian.....	31
D.1. Analisis Kasus Uji 1.....	31
D.2. Analisis Kasus Uji 2.....	31
D.3. Analisis Kasus Uji 3.....	31
D.4. Analisis Kasus Uji 4.....	32
D.5. Analisis Kasus Uji 5.....	32
D.6. Analisis Kasus Uji 6.....	32
D.7. Analisis Kasus Uji 7.....	32
D.8. Analisis Kasus Uji 8.....	32
BAB 5	
KESIMPULAN DAN REFLEKSI.....	33
A. Kesimpulan.....	33
B. Refleksi.....	33
LAMPIRAN.....	34
DAFTAR PUSTAKA.....	35

BAB 1

DESKRIPSI TUGAS

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.

Pattern matching merupakan teknik penting dalam sistem identifikasi sidik jari. Teknik ini digunakan untuk mencocokkan pola sidik jari yang ditangkap dengan pola sidik jari yang terdaftar di database. Algoritma pattern matching yang umum digunakan adalah Bozorth dan Boyer-Moore. Algoritma ini memungkinkan sistem untuk mengenali sidik jari dengan cepat dan akurat, bahkan jika sidik jari yang ditangkap tidak sempurna.

Dengan menggabungkan teknologi identifikasi sidik jari dan pattern matching, dimungkinkan untuk membangun sistem identifikasi biometrik yang aman, handal, dan mudah digunakan. Sistem ini dapat diaplikasikan di berbagai bidang, seperti kontrol akses, absensi karyawan, dan verifikasi identitas dalam transaksi keuangan.

Pada tugas ini, diimplementasikan sistem yang dapat melakukan identifikasi individu berbasis biometrik dengan menggunakan sidik jari. Metode yang akan digunakan untuk melakukan deteksi sidik jari adalah Boyer-Moore dan Knuth-Morris-Pratt. Selain itu, sistem ini akan dihubungkan dengan identitas sebuah individu melalui basis data sehingga terbentuk sebuah sistem yang dapat mengenali identitas seseorang secara lengkap hanya dengan menggunakan sidik jari.

BAB 2

LANDASAN TEORI

A. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma KMP atau Knuth Morris Pratt cukup mirip dengan algoritma brute force. Seperti algoritma brute force, algoritma KMP mencari pattern di dalam string dalam urutan kiri ke kanan. Namun, ketika terjadi mismatch, algoritma KMP akan menggeser (shift) pattern dengan lebih cerdas, yaitu dengan menggeser pattern sedemikian rupa sehingga tidak terjadi wasteful comparison. Hal ini dilakukan dengan mencari panjang prefix terbesar pattern yang sama dengan suffix pattern dari posisi sebelum mismatch (index mismatch - 1). Fungsi pencarian ini disebut dengan Fungsi Pinggiran KMP atau KMP Border Function. Misalnya, apabila terjadi mismatch di index 5, maka akan dicari fungsi pinggiran 4. Apabila nilai fungsi pinggiran 4 adalah 2, maka pencarian akan diulang dari index 2; bukan dari index 0.

B. Algoritma Boyer-Moore (BM)

Algoritma BM atau Boyer Moore adalah algoritma pattern matching yang didasarkan pada teknik *looking-glass* dan teknik *character-jump*. Algoritma BM memeriksa pattern dari belakang (kanan ke kiri). Ketika terjadi *mismatch*, misalnya dengan karakter x pada string, algoritma BM akan mencoba menggeser pattern ke kanan sedemikian rupa sehingga karakter x pada string tersebut sejajar dengan kemunculan karakter x terakhir pada pattern. Apabila karakter x terdapat di dalam pattern, tetapi tidak mungkin disejajarkan dengan menggeser pattern ke kanan, maka pattern akan digeser 1 karakter ke kanan. Apabila karakter x tidak ada di dalam pattern, algoritma akan memeriksa ulang pattern dari awal dengan karakter berikutnya pada string.

C. Regex

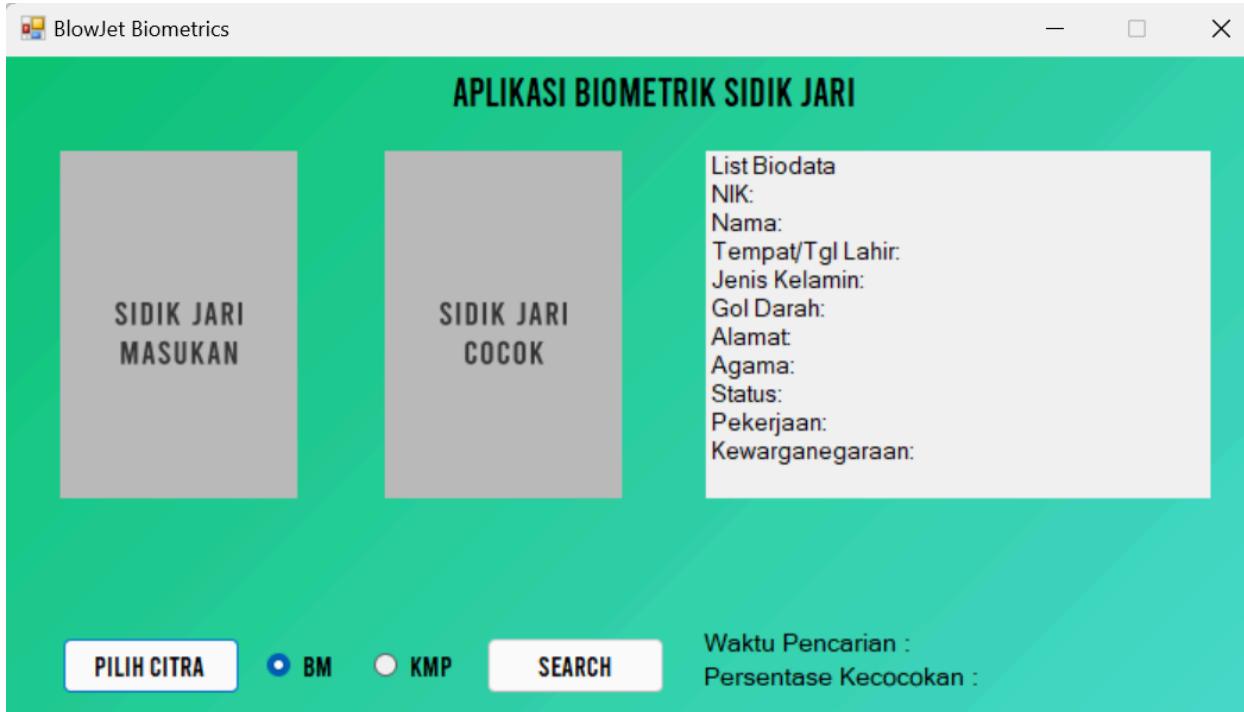
Regex adalah metode pattern matching dengan regular expression. Dengan regex, kita dapat mencari string dengan pola tertentu. Pola tersebut didefinisikan melalui simbol-simbol yang ditetapkan dalam algoritma regex. Contohnya, simbol “[]” menandakan pencarian karakter yang berada di dalam simbol tersebut. Simbol “*” menandakan karakter dengan jumlah yang tidak ditentukan (nol sampai tidak hingga). Dengan penggabungan simbol-simbol tersebut, dapat dicari string dengan pola tertentu. Contohnya, pola regex [A-Z][a-z]* akan mencari string yang dimulai dengan alfabet huruf besar yang dilanjutkan dengan nol atau banyak huruf kecil.

D. Pengukuran Persentase Kemiripan

Apabila pattern matching gagal, maka diperlukan algoritma lain untuk mengukur persentase kemiripan citra/gambar. Salah satu algoritma yang dapat digunakan adalah Hamming Distance. Dalam kasus perbandingan dua string, Hamming Distance diukur dengan menghitung jumlah

karakter berbeda diantara dua string tersebut. Nilai Hamming Distance tersebut kemudian dapat dibagi dengan panjang string untuk mendapatkan persentase kemiripan.

E. Penjelasan Aplikasi Desktop yang Dibangun



Aplikasi yang dibangun terdiri atas sebuah menu yang memungkinkan pengguna untuk mencari biodata individu dari sebuah database berdasarkan citra sidik jari. Aplikasi terdiri atas tombol upload citra sidik jari beserta displaynya yang dapat digunakan pengguna untuk mengupload citra sidik jari pilihannya. Aplikasi juga terdiri atas tombol BM/KMP yang dapat digunakan pengguna untuk memilih algoritma pencarian. Terakhir, aplikasi memiliki tombol search yang apabila ditekan, akan memulai pencarian. Hasil dari pencarian adalah display sidik jari yang paling cocok dari database, list biodata dari individu yang terkait, serta waktu pencarian dan persentase kecocokan input citra sidik jari dengan citra sidik jari yang ditemukan.

BAB 3

ANALISIS PEMECAHAN MASALAH

A. Langkah-Langkah Pemecahan Masalah

Pertama-tama pengguna mengupload gambar sidik jari yang akan dicocokkan dengan menekan tombol “pilih sidik jari”. Program akan menampilkan dialog dimana pengguna dapat memilih file gambar dari folder lokal. Kemudian, pengguna dapat memilih algoritma yang akan digunakan untuk proses pencarian dengan menekan tombol BM atau tombol KMP. Untuk memulai pencarian, pengguna dapat menekan tombol “Search”. Ketika pencarian dimulai, program kemudian akan mengkonversi gambar yang sudah diupload menjadi data ascii. Program lalu mengambil 30 karakter ascii (30 pixel) yang akan dijadikan pattern untuk proses pencocokan. Program kemudian akan menerapkan pattern matching menggunakan algoritma yang dipilih dengan gambar-gambar yang ada di dalam database. Karena database hanya menyimpan *path* menuju file gambar, seperti gambar pilihan sidik jari, program harus terlebih dahulu mengambil file gambar lalu mengkonversinya menjadi ascii. Apabila tidak ditemukan gambar yang sesuai dengan pattern matching, program akan menggunakan Hamming Distance untuk mencari gambar yang paling cocok. Setelah didapatkan gambar yang paling cocok beserta nama individunya, program akan mencoba mencari biodata individu di dalam database. Untuk itu, program akan menggunakan regex untuk mencari nama yang sesuai dengan nama yang telah didapatkan. Apabila ditemukan nama yang cocok, biodata individu akan diambil dan ditampilkan di dalam aplikasi beserta gambar citra sidik jari yang cocok, waktu pencarian, dan persentase kemiripan gambar.

B. Proses Penyelesaian Solusi dengan algoritma KMP dan BM

B.1. KMP

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma yang digunakan untuk pencocokan string atau pola dalam teks. Algoritma ini meningkatkan efisiensi pencarian dibandingkan dengan algoritma brute force dengan cara menghindari perbandingan yang tidak perlu. Prinsip Dasar Algoritma KMP adalah melakukan pencarian pola dalam teks dari kiri ke kanan, seperti algoritma brute force, namun dengan cara yang lebih cerdas dalam menggeser pola saat terjadi ketidaksesuaian.

Terdapat Fungsi Pinggiran (Border Function) dalam Algoritma KMP. Algoritma KMP menggunakan fungsi pinggiran (border function) untuk menentukan seberapa jauh pola harus digeser saat terjadi ketidaksesuaian. Fungsi pinggiran ini dihitung sebelum proses pencocokan dimulai.

- Prefix: Substring dari awal string hingga indeks tertentu.
- Suffix: Substring dari indeks tertentu hingga akhir string.

- Border: Prefix yang juga merupakan suffix.

Langkah-langkah dalam Algoritma KMP

1. Preprocessing Pola (Pattern Preprocessing)

Hitung fungsi pinggiran untuk pola yang diberikan. Fungsi pinggiran (sering disebut juga sebagai fungsi gagal atau fail function) menyimpan ukuran prefix terbesar dari pola yang juga merupakan suffix.

2. Pencocokan Pola dalam Teks

- Pencocokan dimulai dari karakter pertama dari teks.
- Jika terjadi ketidaksesuaian antara teks dan pola pada posisi j , maka geser pola berdasarkan nilai dari fungsi pinggiran $b[j-1]$.
- Proses pencocokan dilanjutkan hingga seluruh teks selesai diperiksa atau pola ditemukan.

Kompleksitas Waktu dalam algoritma KMP.

- Preprocessing: $O(m)$, di mana m adalah panjang pola.
- Pencocokan: $O(n)$, di mana n adalah panjang teks.
- Kompleksitas total: $O(m + n)$, lebih efisien dibandingkan brute force yang $O(mn)$.

Kelebihan KMP

- Tidak perlu kembali ke awal teks saat terjadi ketidaksesuaian, sehingga sangat efisien dalam pemrosesan teks yang sangat besar.

Kekurangan KMP

- KMP menjadi kurang efisien saat ukuran alfabet besar karena peluang ketidaksesuaian lebih besar.

B.2. BM

Algoritma Boyer-Moore adalah salah satu algoritma pencocokan string yang paling efisien. Algoritma ini memanfaatkan dua teknik utama untuk menghindari perbandingan yang tidak perlu, yaitu teknik kaca belakang (looking-glass technique) dan teknik loncatan karakter (character-jump technique).

1. Teknik Kaca Belakang (Looking-Glass Technique)

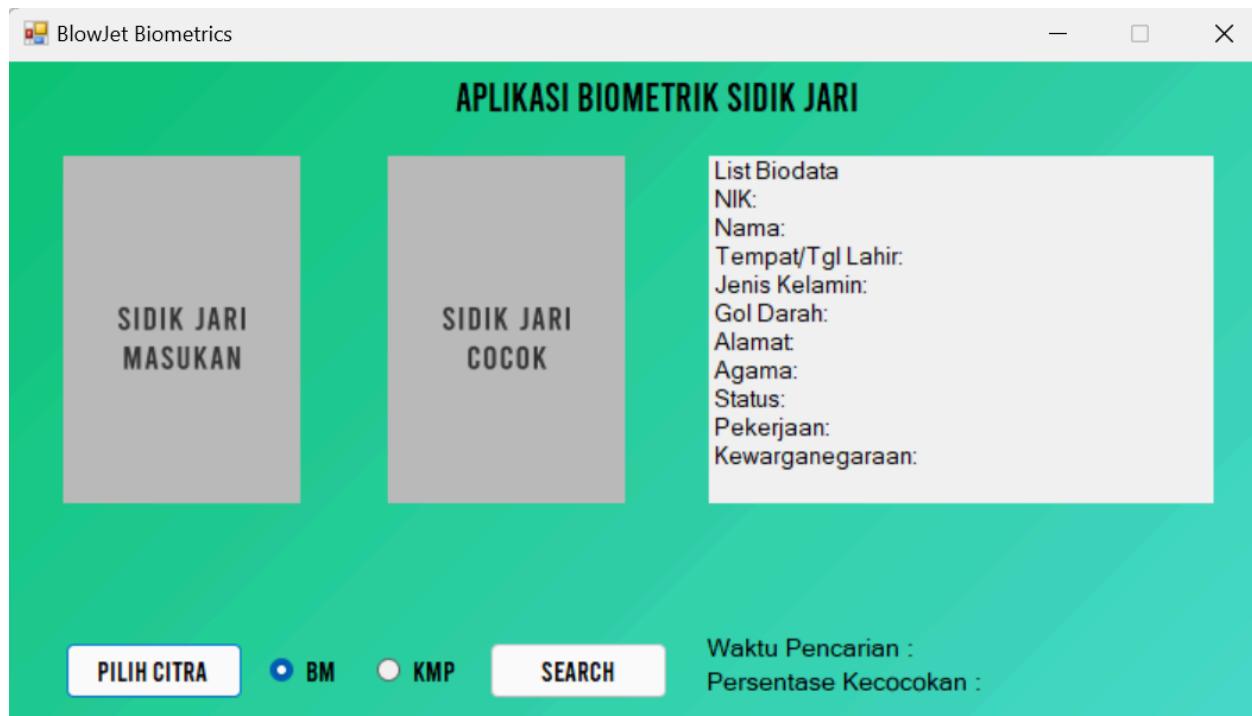
Pada teknik ini, pencarian pola dalam teks dilakukan dari belakang pola ke depan. Artinya, mulai dari karakter paling akhir dari pola dan mencocokkannya dengan teks. Jika terjadi ketidakcocokan, pola digeser untuk mengoptimalkan pencarian berikutnya.

2. Teknik Loncatan Karakter (Character-Jump Technique)

Teknik ini memanfaatkan informasi tentang posisi terakhir dari karakter yang tidak cocok dalam pola. Terdapat tiga kasus yang dipertimbangkan saat terjadi ketidakcocokan:

- Kasus 1: Jika karakter yang tidak cocok terdapat di dalam pola, geser pola ke kanan sehingga kemunculan terakhir dari karakter tersebut dalam pola sejajar dengan karakter di teks.
- Kasus 2: Jika karakter yang tidak cocok ada di pola tetapi tidak bisa diatur sesuai kemunculan terakhir, geser pola ke kanan sebanyak satu karakter.
- Kasus 3: Jika karakter yang tidak cocok tidak ada dalam pola, geser pola sehingga awal pola sejajar dengan karakter berikutnya di teks.

C. Fitur Fungsional dan Arsitektur Aplikasi

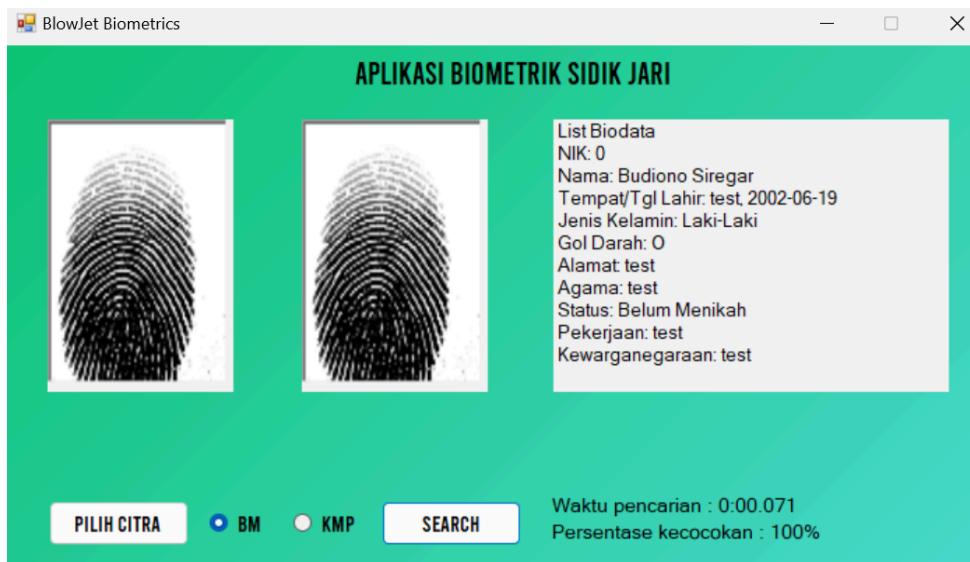


Aplikasi terdiri atas judul aplikasi, tombol “pilih sidik jari”, tombol BM dan KMP, tombol “search”, serta display sidik jari masukan, display sidik jari cocok yang ditemukan, text berupa list biodata individu yang ditemukan, waktu pencarian, dan persentase kecocokan. Pengguna dapat memilih citra sidik jari yang akan dicari dengan menekan tombol “pilih citra” dan program akan menampilkan citra tersebut di display sidik jari masukan. Pengguna juga dapat memilih algoritma pattern matching yang digunakan, antara BM atau KMP, dengan menekan tombol “BM” atau tombol “KMP”. Untuk memulai pencarian, pengguna dapat menekan tombol “Search” yang akan memulai pencarian sidik jari menggunakan algoritma yang telah dipilih. Setelah

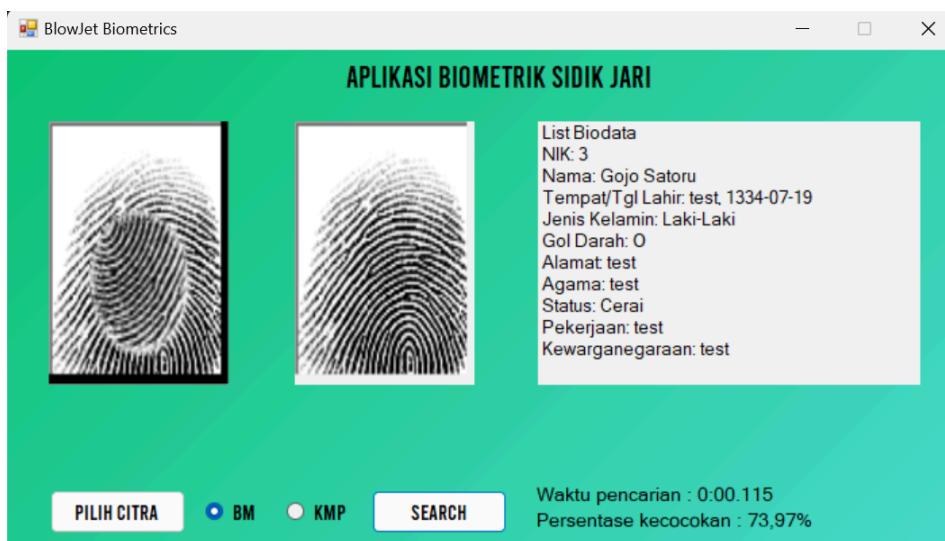
pencarian selesai, program akan menampilkan citra sidik jari yang paling cocok dari database di dalam display sidik jari cocok. Program juga akan menampilkan list biodata dari individu yang bersangkutan beserta waktu pencarian dan persentase kecocokan.

D. Contoh Ilustrasi Kasus

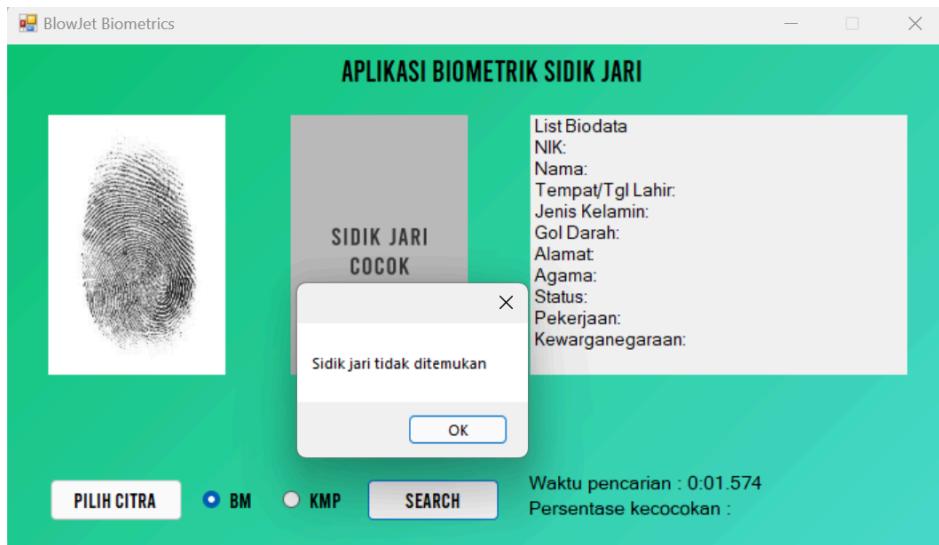
D.1. Kasus 1



D.2. Kasus 2



D.3. Kasus 3



BAB 4

IMPLEMENTASI DAN PENGUJIAN

A. Spesifikasi Teknis Program (struktur data, fungsi, dan prosedur yang dibangun)

A.1. Kelas MainForm

Kelas MainForm adalah kelas yang menjadi pengendali utama GUI program. MainForm merupakan kelas turunan dari kelas System.Windows.Forms.Form. MainForm hanya memiliki 1 atribut baru, yaitu *private string opsi_pencarian* yang berfungsi menyimpan pilihan algoritma (BM/KMP). MainForm antara lain terdiri atas label judul, picturebox untuk sidik jari pilihan, picturebox untuk sidik jari yang cocok, textbox untuk menampilkan biodata individu, label waktu pencarian, label persentase kecocokan, serta 4 tombol, yaitu button Pilih Citra, radiobutton BM, radiobutton KMP, dan button Search. Command yang dipanggil ketika masing-masing tombol ditekan akan dijabarkan sebagai berikut:

Tombol yang ditekan	Source Code	Keterangan
Pilih Citra	<pre>private void pilihButton_Click(object sender, EventArgs e) { String imageLocation = ""; try { OpenFileDialog openFileDialog = new OpenFileDialog(); openFileDialog.CheckFileExists = true; openFileDialog.AddExtension = true; openFileDialog.Multiselect = true; openFileDialog.Filter = "Image files (*.JPG;*.JPEG;*.PNG;*.BMP) *.JPG;*.JPEG; *.PNG;*.BMP"; if (openFileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK) { imageLocation = openFileDialog.FileName; } }</pre>	Ketika tombol Pilih Citra ditekan, pertama-tama, program akan membuat OpenFileDialog baru dimana pengguna dapat memilih file gambar sidik jari. OpenFileDialog tersebut hanya dapat menerima file dalam extension JPG, JPEG, PNG, atau BMP. Apabila pemilihan file gambar berhasil, program akan mengambil lokasi/ <i>path</i> dari file pilihan dan mengubah gambar pada picturebox untuk menampilkan sidik jari pilihan.

	<pre> Console.WriteLine("Input " + imageLocation); pictureBox2.ImageLocation = imageLocation; } } catch (Exception) { MessageBox.Show("Error"); } } </pre>	
BM	<pre> private void radioButton1_CheckedChanged(object sender, EventArgs e) { this.opsi_pencarian = "BM"; // MessageBox.Show(this.opsi_pencarian); } </pre>	Ketika radiobutton BM ditekan, program akan mengubah nilai atribut opsi_pencarian pada MainForm menjadi "BM".
KMP	<pre> private void radioButton2_CheckedChanged(object sender, EventArgs e) { this.opsi_pencarian = "KMP"; // MessageBox.Show(this.opsi_pencarian); } </pre>	Ketika radiobutton KMP ditekan, program akan mengubah nilai atribut opsi_pencarian pada MainForm menjadi "KMP".
Search	<pre> private void button1_Click(object sender, EventArgs e) { if (pictureBox2.ImageLocation is null) { MessageBox.Show("Citra Sidik Jari belum dipilih"); return; } reset_Display(); var timer = new Stopwatch(); timer.Start(); (String, String, double) result = ("", "", 0.0); result = Converter.selectBerkasFromFingerprint(pictur </pre>	Ketika tombol Search ditekan, pertama-tama, program akan memeriksa apakah sudah ada masukan sidik jari yang ingin dicari. Apabila belum ada masukan sidik jari, proses pencarian akan di-abort dan program menampilkan error message. Apabila sudah ada masukan sidik jari, program akan menyalakan stopwatch dan memanggil fungsi "selectBerkasFromFingerprint" dari kelas Converter. Parameter fungsi tersebut

	<pre> eBox2.ImageLocation, this.opsi_pencarian); timer.Stop(); TimeSpan elapsed = timer.Elapsed; String waktu = elapsed.ToString(@"m\:ss\.\fff"); label1.Text = "Waktu pencarian : " + waktu; // if tidak ada yang cukup cocok if (result.Item3 <= 0.70) { MessageBox.Show("Sidik jari tidak ditemukan"); return; } String persentase = (Math.Truncate(result.Item3 * 10000) / 100).ToString() + "%"; label2.Text = "Persentase kecocokan : " + persentase; pictureBox1.ImageLocation = result.Item1; //NIK, Nama,Tempat/Tgl,Kelamin,Darah, Alamat, Agama, Status, Kerja, Negara (String, String, String, String, String, String, String, String, String) biodata = search_Biodata(result.Item2); textBox1.Text = " List Biodata\r\n " + "NIK: " + biodata.Item1 + "\r\n " + "Nama: " + biodata.Item2 + "\r\n " + "Tempat/Tgl Lahir: " + biodata.Item3 + "\r\n " + "Jenis Kelamin: " + biodata.Item4 + "\r\n " + "Gol Darah: " + biodata.Item5 + "\r\n " + "Alamat: " + biodata.Item6 + "\r\n " + + "Agama: " + biodata.Item7 + "\r\n " + </pre>	<p>adalah <i>path</i> file gambar sidik jari masukan (pictureBox2.ImageLocation) dan algoritma yang akan digunakan (this.opsi_pencarian). Setelah pencarian selesai, program akan menghentikan stopwatch dan menampilkan durasi waktu. Apabila ditemukan gambar dengan persentase kecocokan yang lebih dari 70%, program akan mencari biodata individu yang terkait di dalam database menggunakan fungsi “search_Biodata”. Setelah itu, program akan menampilkan biodata individu tersebut di dalam textbox. Apabila tidak ditemukan gambar dengan persentase kecocokan yang lebih dari 70%, program akan menampilkan message "Sidik jari tidak ditemukan".</p>
--	--	---

	<pre> "Status: " + biodata.Item8 + "\r\n" + "Pekerjaan: " + biodata.Item9 + "\r\n" " "Kewarganegaraan: " + biodata.Item10; Console.WriteLine("Finished"); } </pre>	
--	---	--

Kelas MainForm juga memiliki beberapa fungsi utilitas lain sebagai berikut.

Fungsi	Source Code	Keterangan
private void reset_Display()	<pre> private void reset_Display() { pictureBox1.ImageLocation = "./assets/cocok.PNG"; label2.Text = "Percentase kecocokan : "; textBox1.Text = " List Biodata\r\n NIK: \r\n Nama:\r\n Tempat/Tgl Lahir:\r\n Jenis Kelamin:\r\n Gol Darah:\r\n" + " Alamat:\r\n Agama:\r\n Status:\r\n Pekerjaan:\r\n Kewarganegaraan:"; } </pre>	Fungsi utilitas untuk membersihkan picturebox sidik jari yang cocok, textbox list biodata, dan label persentase kecocokan.
private (String, String, String, String, String, String, String, String, String, String) search_Biodata(String nama)	<pre> private (String, String, String, String, String, String, String, String, String, String) search_Biodata(String nama) { //NIK, Nama,Tempat/Tgl,Kelamin,Darah, Alamat,Agama, Status, Kerja, Negara (String, String, String, String, String, String, String, String, String, String) biodata = ("", "", "", "", "", "", "", "", "", "", ""); using (var connection = new SQLiteConnection("Data Source=db/test.db;Version=3;Journal Mode=Off", true)) { connection.Open(); } } </pre>	Fungsi utilitas untuk mencari individu di dalam database dengan nama yang cocok dengan parameter “nama”, lalu mengembalikan biodatanya di dalam bentuk tuple. Pertama-tama, program akan menjalankan query ke database untuk mengambil semua biodata. Untuk setiap data, program akan memperbaiki biodata dengan mengubah angka-angka menjadi karakter yang benar. Lalu, program akan

	<pre> // Define the SQL query string query = "SELECT * FROM biodata"; // Create a command using (var command = new SQLiteCommand(query, connection)) { // Execute the query and get a data reader using (var reader = command.ExecuteReader()) { // Read the data while (reader.Read()) { string name = reader.GetString(reader.GetOrdinal("na ma")); /* Console.WriteLine("Test"); Console.WriteLine(nama); Console.WriteLine(name);*/ // input = name, pattern = nama String patternAlay = "[0123456][a-z]"; string input = Regex.Replace(name, patternAlay, new MatchEvaluator(StringConversion.Prog ram.replacement)); String patternSingkat = "[aiueo]"; string pattern = Regex.Replace(nama, patternSingkat, match => match+"?"); /* Console.WriteLine(input); Console.WriteLine(pattern);*/ </pre>	<p>memodifikasi parameter “nama” menjadi pola regex supaya dapat <i>match</i> dengan nama yang disingkat-singkat. Terakhir, program akan menjalankan Regex.IsMatch untuk menerapkan regex dengan input berupa nama dari database yang sudah diperbaiki dan pattern berupa parameter nama yang sudah dimodifikasi. Apabila timbul match, fungsi akan mengembalikan tuple berisi biodata. Jika tidak ada match, fungsi akan mengembalikan tuple kosong.</p>
--	--	---

```

        bool matchNama =
Regex.IsMatch(input, pattern);

        if (matchNama)
{
    string capitalize =
Regex.Replace>Nama.ToLower(),
@"\b[a-z]", match =>
match.Value.ToUpper());

/*Console.WriteLine("Match!");

Console.WriteLine(capitalize);

Console.WriteLine(matchNama);*/

        String nik =
reader.GetString(reader.GetOrdinal("NI
K"));

        String tempat =
reader.GetString(reader.GetOrdinal("te
mpat_lahir"));

        String tanggal =
reader.GetString(reader.GetOrdinal("ta
nggal_lahir"));

        String kelamin =
reader.GetString(reader.GetOrdinal("je
nis_kelamin"));

        String darah =
reader.GetString(reader.GetOrdinal("go
longan_darah"));

        String alamat =
reader.GetString(reader.GetOrdinal("al
amat"));

        String agama =
reader.GetString(reader.GetOrdinal("ag
ama"));

        String status =
reader.GetString(reader.GetOrdinal("sta
tus_perkawinan"));

        String pekerjaan =
reader.GetString(reader.GetOrdinal("pe
kerjaan"));

        String kewarganegaraan
=

```

	<pre> reader.GetString(reader.GetOrdinal("ke warganegaraan")); biodata = (nik, capitalize, tempat + ", " + tanggal, kelamin, darah, alamat, agama, status, pekerjaan, kewarganegaraan); return biodata; } } } /*Console.WriteLine("Here"); foreach ((String, String) s in temp) { Console.WriteLine(s.Item1, s.Item2); } */ return biodata; } </pre>	
--	---	--

A.2. Kelas Converter

Kelas Converter merupakan kelas utama yang bertugas untuk memproses fingerprint yang diinput oleh user dan mencari fingerprint yang paling cocok di database. Semua metode di kelas ini merupakan static method, dimana kelas ini juga tidak memiliki atribut khusus. Beberapa method dari kelas ini adalah sebagai berikut:

Method	Source code	Keterangan
public static (String, String, double) selectBerk asFromFing erprint(St ring imagePath,	// ini ngeconvert ke 30 ascii querinya String asciiFull = ImageToBinaryString(imagePa th); String asciiQuery = FindMostUniqueSubstring(asc iiFull); // buat result (image path yang akan dipilih terakhir,	Method ini merupakan method utama dari kelas ini. Method ini berfungsi untuk menerima path dari image yang diinput user dan algoritma yang dipilih user. Nantinya program pertama akan mengambil seluruh atribut relasi sidik_jari pada database,

```
String  
algo)
```

```
nantinya jadi nama  
orangnya)  
(String, String) result =  
("", "");  
// buat persentase  
kecocokan  
double cocok = 0;  
// temporary, harusnya list  
of `berkas_citra` di sql  
nya  
List<(String, String)> a =  
DatabaseManager.GetSidikJar  
i();  
// coba pake kmp/bm dulu  
foreach ((String, String)  
imageItr in a){  
    if (algo == "KMP" &&  
KMPString.KMPmatching(Image  
ToBinaryString(imageItr.Ite  
m1), asciiQuery) != -1){  
        result = imageItr;  
        cocok = 1;  
    } else if (algo == "BM"  
&&  
BoyerMooreString.Search(I  
mageToBinaryString(imageItr.I  
tem1), asciiQuery) != -1){  
        result = imageItr;  
        cocok = 1;  
    }  
}  
  
// hamming distance  
if (result.Item1 == "")  
{  
    int minDist = -1;  
    foreach ((String,  
String) imageItr in a)  
    {  
        String temp =  
ImageToBinaryString(imageIt  
r.Item1);  
  
        int res =  
hammingDistance(temp,  
asciiFull);
```

lalu memasukkannya ke sebuah array of tuple. Lalu program memulai pencocokan. Setelah ditemukan yang paling cocok, program mereturn path dari database yang cocok, nama (nama rusak) dari orang tersebut, dan persentase kecocokannya (dalam double, range 0-1)

	<pre> if (minDist == -1 res < minDist) { minDist = res; result = imageItr; Double maxSize = Math.Max(asciiFull.Length, result.Item1.Length); cocok = ((maxSize - minDist) / maxSize); } } return (result.Item1, result.Item2, cocok); } } </pre>	
static String ImageToBin aryString(string imagePath)	<pre> Image<Rgba32> image = Image.Load<Rgba32>(imagePat h); String binaryTemp = ""; for (int i = 0; i < image.Height; i++) { for (int j = 0; j < image.Width; j++) { Rgba32 pixelColor = image[j, i]; int grayscaleValue = Convert.ToInt32(pixelColor. R * 0.299f + pixelColor.G * 0.587f + pixelColor.B * 0.114f); binaryTemp += Convert.ToString((char)gray scaleValue); } } return binaryTemp; </pre>	Method ini merupakan method pendukung yang berfungsi untuk membuka image dari imagepath yang diinput, dan mendapat grayscale tiap pixelnya (diambil m x n pixel sesuai gambar). Nantinya nilai grayscale tiap pixel gambar akan diubah menjadi ASCII, lalu diconcat menjadi satu, dan akhirnya direturn.
public static	Dictionary<char, int> charCount = new	Method ini merupakan method yang berguna

```

string
FindMostUn
iqueSubstr
ing(string
input)
{
    Dictionary<char, int> charCount;
    int maxUnique = 0;
    int bestI = 0;

    for (int i = 0; i < 30; i++) {
        if (!charCount.ContainsKey(input[i])) charCount[input[i]] = 0;

        charCount[input[i]]++;
    }
    maxUnique = charCount.Count;

    for (int i = 30; i < input.Length; i++) {
        char outgoingChar = input[i - 30];

        charCount[outgoingChar]--;
        if (charCount[outgoingChar] == 0)
            charCount.Remove(outgoingChar);

        char incomingChar = input[i];
        if (!charCount.ContainsKey(incomingChar))
            charCount[incomingChar] = 0;

        charCount[incomingChar]++;
        if (charCount.Count > maxUnique) {
            maxUnique = charCount.Count;
            bestI = i;
        }
    }
}

```

untuk mencari substring dengan panjang 30 paling unik. Nantinya substring itu akan direturn dan dimanfaatkan untuk algoritma KMP/BM sebagai pattern.

	<pre> i - 30 + 1; } } return input.Substring(bestI, 30); </pre>	
	<pre> private static int hammingDis tance(Stri ng str1, String str2) { int count = Math.Abs(str1.Length - str2.Length); int n = Math.Min(str1.Length, str2.Length); for (int i = 0; i < n; i++) { if (str1[i] != str2[i]) { count++; } } return count; } </pre>	Method ini merupakan aplikasi dari hamming distance, yaitu algoritma untuk mencari jarak antara dua string.

A.3. Kelas Booyer Moore

Kelas ini merupakan Kelas yang mengimplementasikan Algoritma Booyer Moore dalam String matching. Kelas ini hanya berisi static method, dan tidak memiliki atribut ataupun method khusus. Berikut merupakan method-method dari kelas Booyer Moore:

Method	Source Code	Keterangan
public static List<int> BuildKMP(S tring arg)	<pre> List<int> temp = new List<int>(); temp.Add(0); int j = 0, i = 1; while (i < arg.Length){ if (arg[j] == arg[i]){ temp.Add(j + 1); i++; j++; } else { if (j == 0){ temp.Add(0); } } } </pre>	Method ini merupakan method pendukung. Method ini membuat tabel border yang nantinya digunakan di method utama KMPString.

	<pre> i++; } else j = temp[j-1]; } } return temp; } </pre>	
public static int KMPmatchin g(String arg, String pattern)	<pre> List<int> temp = BuildKMP(pattern); int i = 0, j = 0; while (i < arg.Length && j < pattern.Length) { if (arg[i] != pattern[j]){ if (j == 0) i++; else j = temp[j-1]; } else{ i++; j++; } } if (j == pattern.Length) return i-j; else return -1; </pre>	Method ini merupakan method utama dari kelas KMPString. Method ini akan melakukan algoritma Knuth-Morris-Pratt dengan menerima argumen berupa pattern dan text, dan nantinya method ini akan mereturn -1 jika tidak ditemukan match atau lokasi indeks awal jika ditemukan match

A.4. Kelas KMPString

Kelas ini merupakan Kelas yang mengimplementasikan Algoritma Knuth-Morris-Pratt dalam String matching. Kelas ini hanya berisi static method, dan tidak memiliki atribut ataupun method khusus. Berikut merupakan method-method dari kelas KMPString:

Method	Source Code	Keterangan
public static List<int> BuildKMP(String arg)	<pre> List<int> temp = new List<int>(); temp.Add(0); int j = 0, i = 1; while (i < arg.Length) { if (arg[j] == arg[i]){ temp.Add(j + 1); i++; j++; } else { if (j == 0){ temp.Add(0); i++; } else j = temp[j-1]; } } </pre>	Method ini merupakan method pendukung. Method ini membuat tabel border yang nantinya digunakan di method utama KMPString.

	<pre> } } return temp; } public static int KMPmatchin g(String arg, String pattern) { List<int> temp = BuildKMP(pattern); int i = 0, j = 0; while (i < arg.Length && j < pattern.Length) { if (arg[i] != pattern[j]){ if (j == 0) i++; else j = temp[j-1]; } else{ i++; j++; } } if (j == pattern.Length) return i-j; else return -1; } </pre>	<p>Method ini merupakan method utama dari kelas KMPString. Method ini akan melakukan algoritma Knuth-Morris-Pratt dengan menerima argumen berupa pattern dan text, dan nantinya method ini akan mereturn -1 jika tidak ditemukan match atau lokasi indeks awal jika ditemukan match</p>
--	---	---

A.5. Kelas DatabaseManager

Kelas DatabaseManager merupakan kelas yang digunakan untuk mendapatkan hal yang diinginkan dari database dengan cara mengquery database. File database yang digunakan berekstensi db (local database, sql harus diconvert ke db terlebih dahulu, penulis menggunakan tools pengguna github [berikut](#)). Berikut merupakan method-method dari kelas DatabaseManager:

Method	Source Code	Keterangan
public static List<(Stri ng, String)> GetSidikJa ri()	<pre> List<(String, String)> temp = new List<(string, string)>(); using (var connection = new SQLiteConnection("Data Source=db/test.db;Version=3 ;")) { connection.Open(); // Define the SQL query string query = "SELECT * FROM sidik_jari"; } </pre>	<p>Method ini merupakan method yang bertujuan untuk mendapatkan seluruh data di relasi sidik_jari di database</p>

```

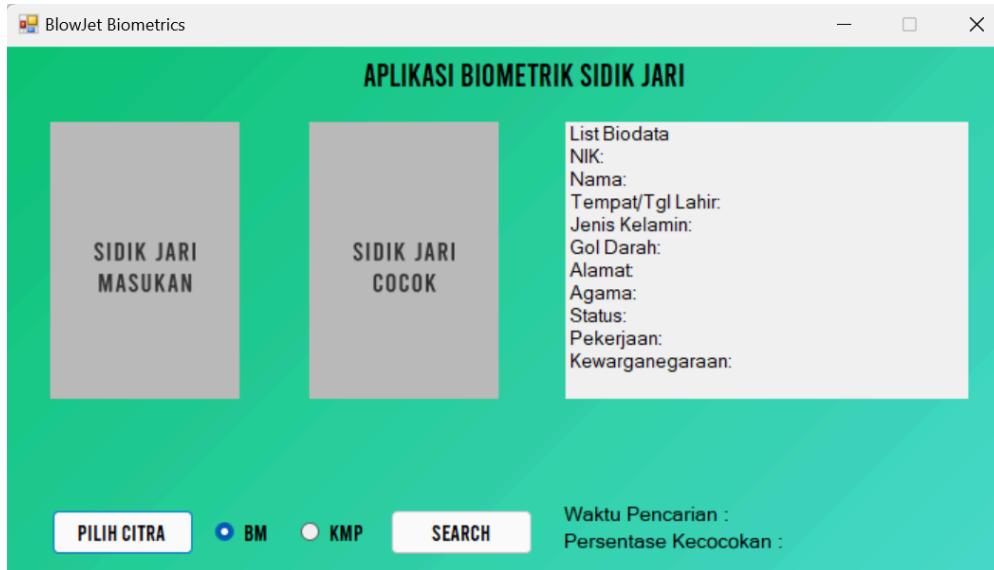
        // Create a command
        using (var command =
new SQLiteCommand(query,
connection))
{
    // Execute the
query and get a data reader
    using (var reader =
command.ExecuteReader())
    {
        // Read the
data
        while
(reader.Read())
        {
            // Example of
reading columns by name
            string berkas =
reader.GetString(reader.GetOrdinal("berkas_citra"));
            string name =
reader.GetString(reader.GetOrdinal("nama"));
            //
Console.WriteLine(berkas +
" " + name);

            temp.Add((berkas, name));
        }
    }
}
return temp;

```

B. Tata Cara Penggunaan Program

Program dapat dijalankan dengan menjalankan file “src.exe” yang terletak di dalam folder src/bin/Release/src.exe. Ketika program dijalankan, program akan memiliki tampilan seperti berikut:



Beberapa fitur program antara lain:

1. Tombol Pilih Citra dan Display Sidik Jari Masukan

Pengguna dapat memberi input citra yang ingin dicari dengan menekan tombol Pilih Citra. Setelah tombol ditekan, program akan menampilkan dialog dimana pengguna dapat memilih file gambar dari folder lokal. Setelah file gambar dipilih, program akan menampilkan gambar di display Sidik Jari Masukan.

2. Tombol BM-KMP

Pengguna dapat memberi input algoritma pattern matching yang ingin digunakan dengan menekan tombol BM atau tombol KMP, sesuai dengan jenis algoritmanya. Program akan otomatis menyesuaikan algoritma yang akan digunakan pada proses pencarian, yaitu ketika tombol Search ditekan.

3. Tombol Search

Pengguna dapat memulai proses pencarian dengan menekan tombol Search. Setelah itu, program akan mencocokkan gambar input dengan gambar yang ada di database. Apabila gambar yang cocok ditemukan, program akan menampilkan display gambar yang cocok, list biodata dari individu yang terkait, serta waktu pencarian dan persentase kecocokan gambar. Apabila tidak ada gambar yang cocok, program akan menampilkan dialog message yang memberi tahu pengguna bahwa sidik jari tidak ditemukan.

4. Display Sidik Jari Cocok, List Biodata, Waktu Pencarian, dan Persentase Kecocokan

Apabila ditemukan sidik jari yang cocok dalam proses pencarian, program akan menampilkan gambar sidik jari tersebut di display Sidik Jari Cocok. Selain itu program

juga akan menampilkan list biodata dari individu yang terkait. Biodata tersebut mengandung NIK, nama, tempat/tanggal lahir, jenis kelamin, golongan darah, alamat, agama, status pernikahan, pekerjaan, dan kewarganegaraan individu. Program juga akan menampilkan durasi waktu pencarian dan persentase kecocokan gambar.

C. Hasil Pengujian

C.1. Kasus Uji 1

Control case, dengan algoritma BM.



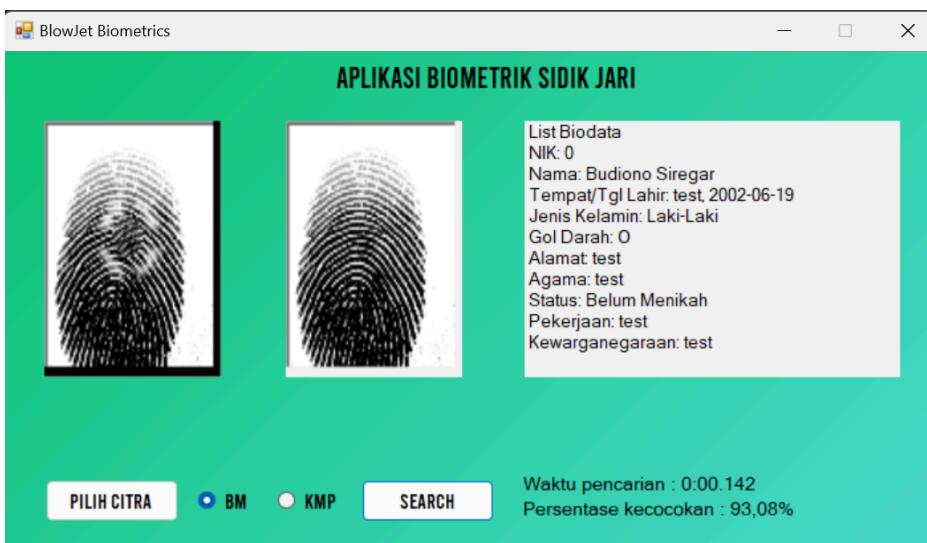
C.2. Kasus Uji 2

Individu dan jari yang sama dengan kasus uji 1, dengan algoritma KMP



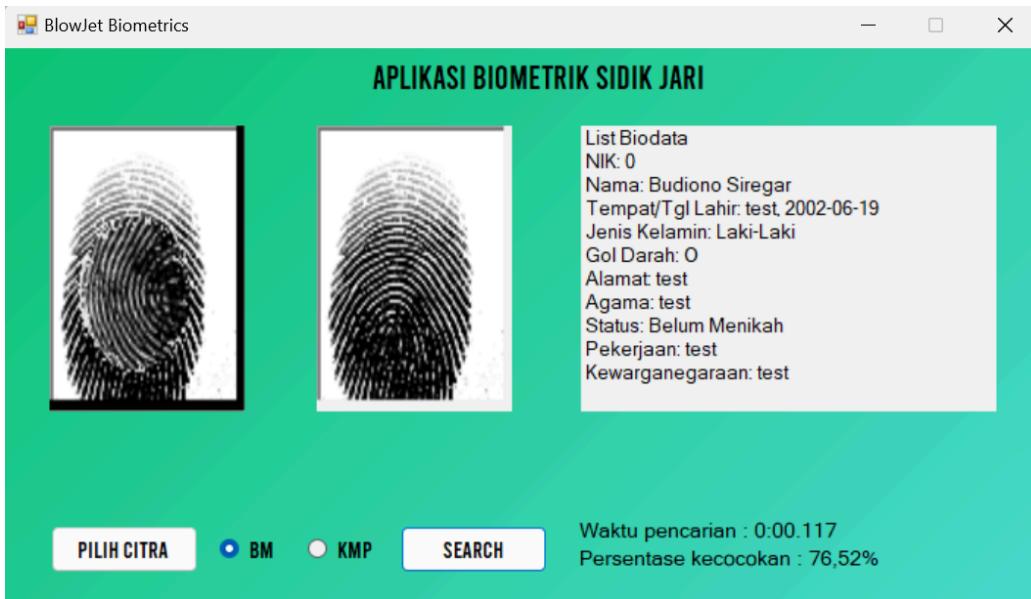
C.3. Kasus Uji 3

Individu dan jari yang sama dengan kasus uji 1, dengan jari yang di-altered.



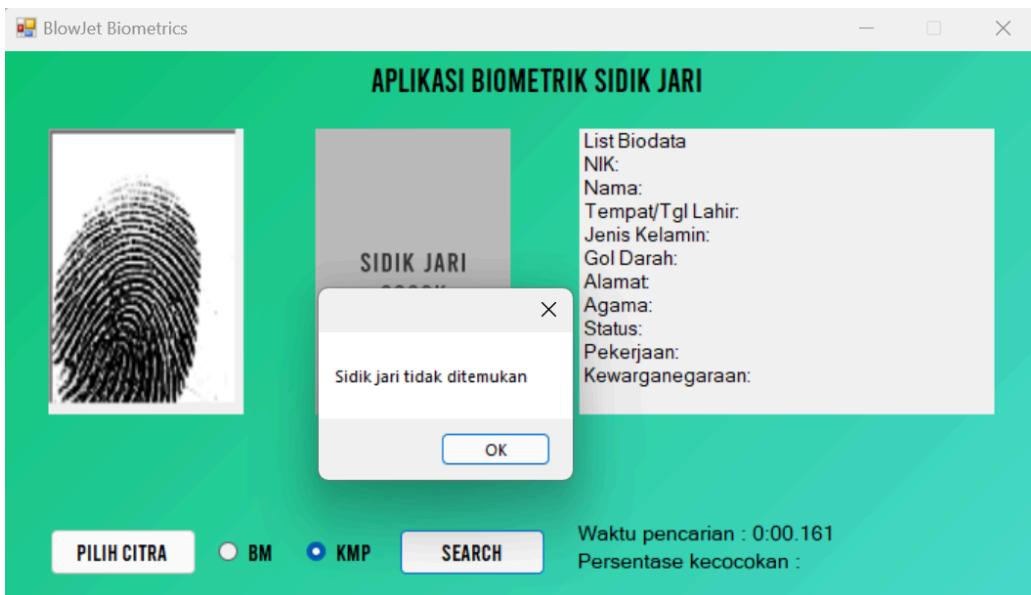
C.4. Kasus Uji 4

Individu dan jari yang sama dengan kasus uji 1, dengan jari yang di-altered secara lebih kasar.



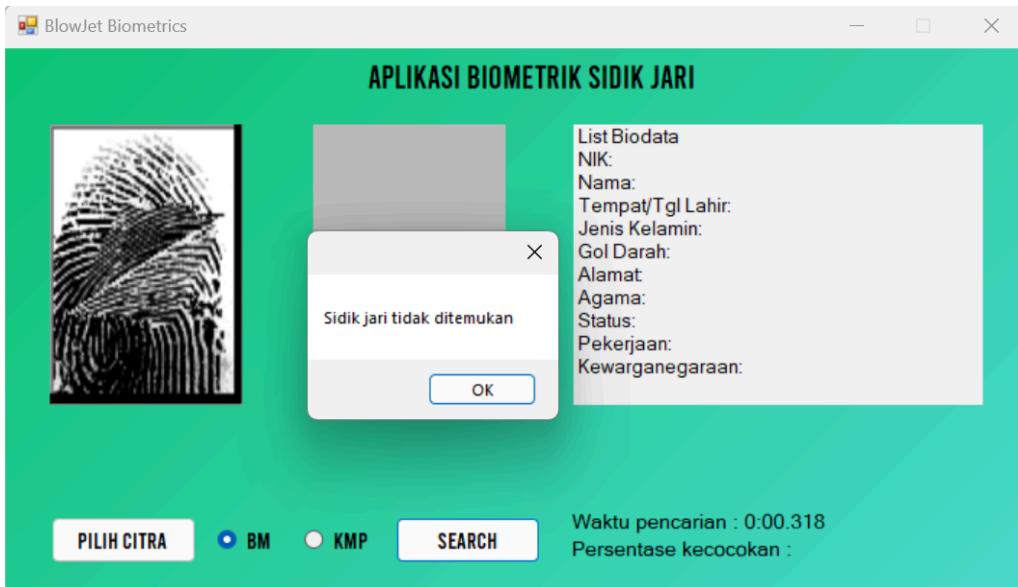
C.5. Kasus Uji 5

Individu yang sama dengan kasus uji 1, tetapi dari jari yang berbeda.



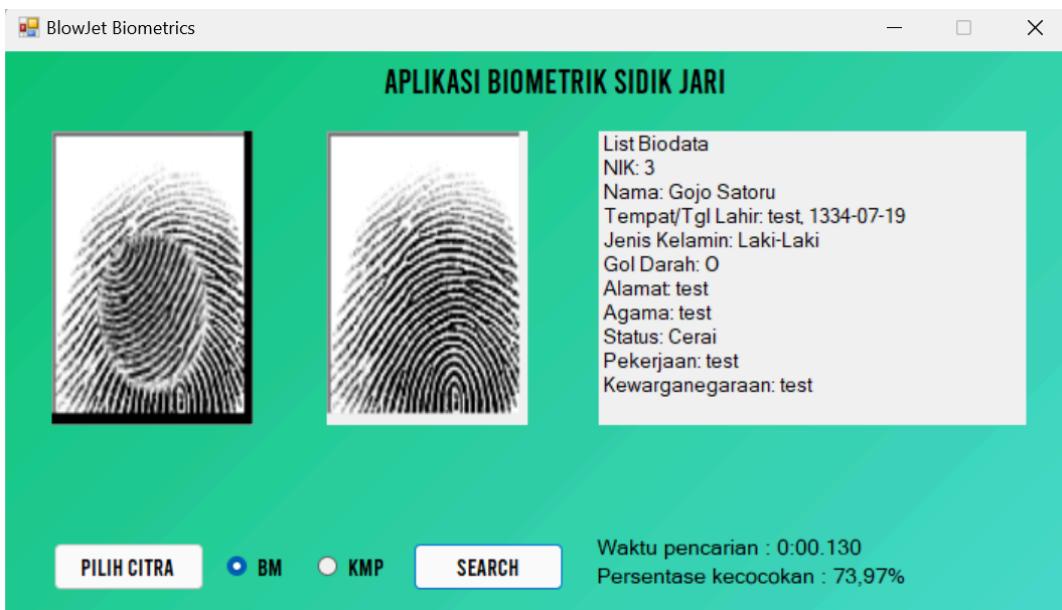
C.6. Kasus Uji 6

Individu yang sama dengan kasus uji 1, tetapi dari jari yang berbeda dan jari tersebut di-altered.



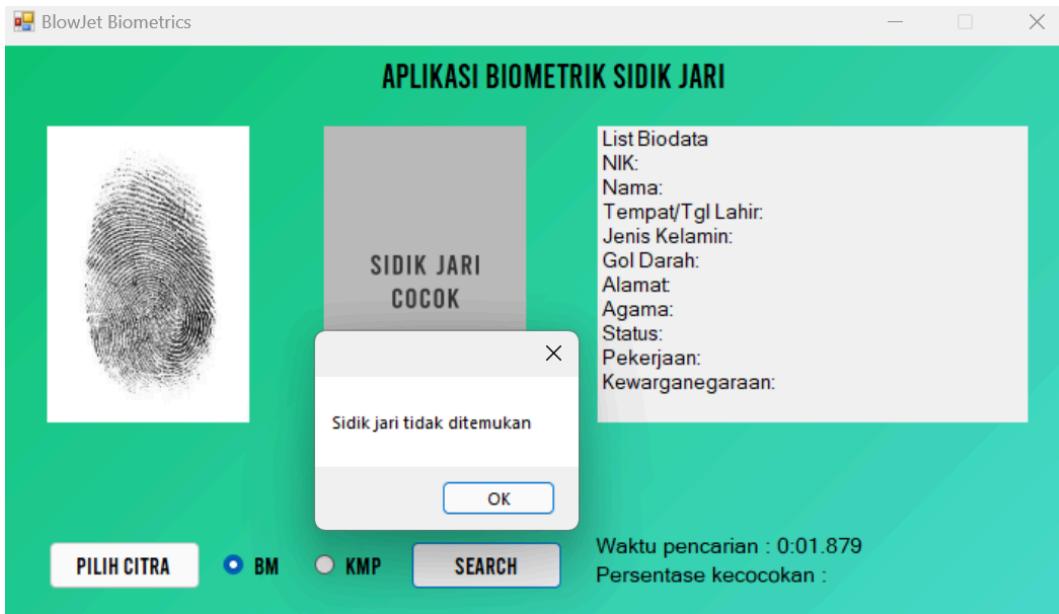
C.7. Kasus Uji 7

Individu berbeda dengan kasus uji 1.



C.8. Kasus Uji 8

Jari dari individu yang belum terdata di dalam database.



D. Analisis Hasil Pengujian

D.1. Analisis Kasus Uji 1

Kasus Uji 1 merupakan kasus dimana fingerprint yang diinput oleh pengguna merupakan salah satu fingerprint yang ada di database. Karena sama persis, maka program mengeluarkan kecocokan 100% yang berarti proses algoritma Boyer-Moore Matching dengan pattern berupa 30 pixel yang paling cocok dan text berupa seluruh pixel gambar di database tidak mereturn -1.

D.2. Analisis Kasus Uji 2

Kasus Uji 2 merupakan kasus yang sama dengan kasus uji 1, akan tetapi di kasus uji 2 digunakan algoritma Knuth-Morris-Pratt Matching yang juga tidak mereturn -1 karena kecocokannya 100%. Akan tetapi, algoritma ini sedikit lebih lambat daripada algoritma Boyer-Moore di kasus ini (terjadi lebih banyak comparison antar karakter di algoritma Knuth-Morris-Pratt).

D.3. Analisis Kasus Uji 3

Kasus Uji 3 merupakan fingerprint dari individu yang sama seperti kasus uji 2 dan 3, namun fingerprint di kasus uji 3 terdapat kerusakan kecil di tengah. Program tidak mereturn 100% karena prosesi KMP/BM gagal (terpilih 30 pixel yang cocok berada di bagian rusak), sehingga program menggunakan algoritma Hamming Distance untuk menentukan seberapa mirip kedua fingerprint tersebut. Akhirnya program mendapatkan 93,08% kemiripan.

D.4. Analisis Kasus Uji 4

Kasus uji 4 merupakan fingerprint dari individu yang sama seperti kasus uji sebelumnya, namun fingerprint di kasus uji ini terdapat kerusakan cukup besar di tengah berupa perputaran fingerprint. Program tidak mereturn 100% karena proses KMP/BM gagal, sehingga program menggunakan algoritma Hamming Distance didapat kemiripan sebesar 76,52%.

D.5. Analisis Kasus Uji 5

Kasus uji 5 merupakan fingerprint dari individu yang sama, tetapi dengan menggunakan jari yang berbeda. Secara teori, fingerprint antar jari-jari seseorang berbeda dengan yang lainnya. Karena di database hanya terdapat jari orang tersebut yang berbeda dari jari yang di input, maka program mengoutput “Sidik jari tidak ditemukan”.

D.6. Analisis Kasus Uji 6

Kasus uji 6 merupakan fingerprint dari individu yang sama dari sebelumnya, akan tetapi selain menggunakan jari yang berbeda, fingerprint ini juga terdapat kerusakan ditengah. Karena rusak dan jari yang berbeda, maka semakin tidak cocok dengan jari yang ada di database. Oleh karena itu, program mengoutput “Sidik jari tidak ditemukan”.

D.7. Analisis Kasus Uji 7

Kasus uji 7 merupakan fingerprint dari individu yang berbeda dari individu sebelumnya. Pengguna memasukkan fingerprint orang tersebut yang terdapat kerusakan di tengah, sehingga program menggunakan hamming distance dan menemukan 73,97% kemiripan antara kedua fingerprint tersebut.

D.8. Analisis Kasus Uji 8

Kasus uji 8 merupakan fingerprint dari individu yang belum terdaftar di database, sehingga tentu saja program tidak menemukan kemiripan lebih dari 70% (batas untuk menganggap kedua fingerprint mirip).

BAB 5

KESIMPULAN DAN REFLEKSI

A. Kesimpulan

Berdasarkan implementasi dan pengujian yang telah dilakukan dalam tugas besar ini, dapat disimpulkan bahwa Algoritma Boyer-Moore (BM) dan Knuth-Morris-Pratt (KMP) efektif digunakan untuk pencocokan pola sidik jari. Namun, Boyer-Moore lebih cepat dalam pencocokan pola dibandingkan KMP pada beberapa kasus uji. Pada pengujian, ketika sidik jari yang diuji mengalami kerusakan, efektivitas algoritma BM dan KMP menurun. Dalam kondisi ini, penggunaan algoritma Hamming Distance menjadi solusi dalam pengukuran persentase kemiripan. Pada pengujian, program akan menunjukkan hasil terbaik jika data sidik jari yang diuji terdapat dalam basis data. Ketidaksesuaian atau kerusakan pada data yang diuji menyebabkan penurunan persentase kecocokan atau bahkan kegagalan dalam pencarian sidik jari. Selain itu, fingerprints dari jari yang berbeda pada individu yang sama menghasilkan ketidaksesuaian yang signifikan, menunjukkan pentingnya konsistensi dalam pengambilan sidik jari yang sama untuk proses identity verification.

B. Refleksi

Melalui tugas besar ini, kami belajar pentingnya pemilihan algoritma yang tepat seperti KMP dan Boyer Moore dalam pemrosesan dan pencocokan pola sidik jari. Kami juga menyadari bahwa data yang berkualitas dan basis data yang lengkap sangat penting dalam sistem identifikasi biometrik. Proses pengembangan program ini memberikan kami wawasan mendalam tentang tantangan yang dihadapi dalam penerapan teknologi biometrik dan mengajarkan kami tentang pentingnya ketelitian dan optimasi dalam pengembangan algoritma. Tugas besar ini juga mendorong kami untuk terus belajar dan mengadaptasi teknologi terbaru untuk mengatasi keterbatasan dan tantangan yang ada, serta memberikan solusi yang lebih baik dan efisien dalam bidang identifikasi biometrik.

LAMPIRAN

Tautan repository:

https://github.com/NameLessAth/Tubes3_Blow-Jet

Tautan video:

https://www.youtube.com/watch?v=YTG4sSpTxCM&ab_channel=EdogawaEduardus

DAFTAR PUSTAKA

Munir, R. (n.d.). Homepage Rinaldi Munir. <https://informatika.stei.itb.ac.id/~rinaldi.munir/>