Aplikasi Representasi Pohon K-D dan Minutiae untuk Menentukan Kecocokan Sidik Jari

Marvin Scifo Y. Hutahaean - 13522110[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522110@std.stei.itb.ac.id

***Abstrak*—Sudah menjadi standar bagi forensik untuk melakukan identifikasi orang-orang dengan melakukan penyocokan sidik jari. Dengan menggunakan 2 komponen yang penting yaitu K-D Tree dan minutiae, kecocokan dari kedua sidik jari bisa ditemukan. Untuk menemukan kecocokan sidik jari, pertama dilakaukan ekstraksi minutiae dari kedua sidik jari, pemasukkan data minutiae dari dataset ke dalam K-D Tree, perbandingan dengan mencari tetangga terdekat, perbandingan ambang dengan jarak Euclidean dari titik minutiae, dan pencarian nilai kesamaan. Hasil yang diharapkan adalah kesamaan nilainya lebih dari atau sama dengan 50%.**

***Keywords*—K-D Tree, Minutiae, Sidik Jari, Kecocokan**

I. PENDAHULUAN

Sidik jari adalah hasil translasi dari kulit telapak manusia yang bentuknya berupa loops, curves, dan swirls. Sidik jari biasanya bisa didapat dengan mencapkan tinta atau bahan-bahan lainnya ke bagian dari tangan atau kaki manusia. Biasanya untuk mengidentifikasi sidik jari, jari tangan adalah bagian tubuh manusia yang biasanya digunakan sebagai sampel. Sidik jari adalah salah satu metode pengenalan kembali identitas akan orang-orang yang ingin dicari. Metode ini sudah menjadi standar di dalam forensik selama bertahun-tahun. Logika dan sains dibelakang identifikasi sidik jari sebenarnya sudah ditemukan sejak zaman Babilonia untuk melakukan transaksi dalam bisnis. Identifikasi sidik jari atau Dactyloscopy adalah ilmu yang mendalami sidik jari dan kecocokannya untuk melakukan pengecekan kembali akan identitas orang dengan cara mengamati garis yang ada pada garis jari tangan dan telapak kaki.

Sidik jari paling sering digunakan di dalam kepolisian untuk menemukan pelaku dari sebuah aksi kriminal. Biasanya di Tempat Kejadian Perkara (TKP), terdapat jejak sidik jari yang ditinggalkan oleh seorang kriminal yang bisa diambil sampelnya oleh tim forensik. Setelah diambil, tim akan melakukan pengecekan dengan data-data sidik jari yang dimiliki oleh masyarakat setempat. Jika terdapat kecocokan, maka kemungkinan besar pelakunya adalah orang dengan sidik jari tersebut.

Pada makalah ini, akan dijelaskan bagaimana cara mengimplementasikan KD-Tree, normalisasi gambar, dan Minutiae dari sebuah sidik jari. Selain itu, berbagai metode akan digunakan untuk melakukan identifikasi. Salah satu metode yang akan digunakan adalah pembuatan program dengan menggunakan Python. Ini akan menjadi salah satu cara untuk mengidentifikasi sebuah sampel sidik jari.

II. LANDASAN TEORI

Pada makalah ini, identifikasi sidik jari akan dicari dengan menggunakan K-Dimensional Tree (K-D Tree). Sebelum membahas (K-D Tree), ada beberapa hal-hal dan teori-teori dasar yang perlu diketahui sebelum teori terkait KD-Tree dibahas.

A. Graf (Graph)

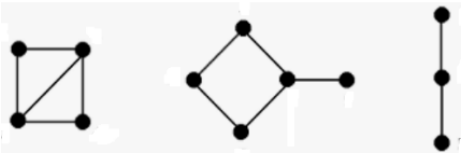
Graf adalah data-data yang merupakan hasil dari representasi objek-objek diskrit dan hubungan objek-objek tersebut terhadap satu sama lain. Pada umumnya, graf terdiri dari objek yang bernama simpul (*vertices*) dan penghubung yang bernama sisi (*edges*). Pada makalah ini, graf yang digunakan adalah graf yang tidak memiliki sirkuit yaitu pohon.

Graf dapat didefinisikan sebagai G, himpunan dari simpul atau *vertices* didefinisikan sebagai V, dan himpunan dari sisi atau *edges* didefinisikan sebagai E. Berdasarkan definisi, graf terdiri dari titik dan garis sehingga bisa dinyatakan dengan G = (V,E). Himpunan simpul atau V terdiri dari kumpulan titik yang bisa dinyatakan sebagai $\{v_1,v_2,\ldots,v_n\}$ dan himpunan sisi atau E terdiri dari kumpulan sisi penghubung simpul yang bisa dinyatakan sebagai $\{e_1,e_2,\ldots,e_n\}$.

Graf memiliki beberapa jenis berdasarkan kategori-kategori yang berbeda. Kategori-kategori tersebut berupa berdasarkan ada tidaknya gelang atau sisi ganda, orientasi arah pada sisi, dll.

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, graf bisa digolongkan menjadi dua jenis yaitu graf sederhana (*simple graph*) dan graf tak sederhana (*unsimple graph*)

1. Graf Sederhana (*Simple Graph*)
   Graf sederhana adalah graf yang tidak mengandung gelang atau sisi yang berganda. Beginilah contoh dari graf sederhana.

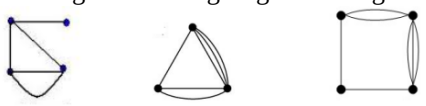


Gambar 2.1 Contoh-contoh graf sederhana

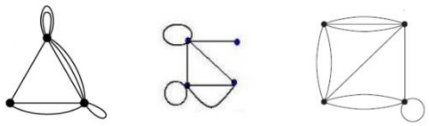
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

2. Graf Tak Sederhana (*Unsimple Graph*)
   Graf tak sederhana adalah jenis graf yang mengandung gelang atau sisi ganda pada strukturnya. Graf tak sederhana memiliki 2 jenis yaitu graf ganda dan graf semu.
   a. Graf ganda
      Graf ganda adalah graf yang mengandung sisi ganda
   b. Graf semu adalah graf yang mengandung sisi gelang

Berikut adalah gambar dari graf ganda dan graf semu.



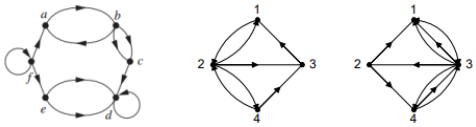
Gambar 2.2 Contoh-contoh graf ganda
Sumber : https://informatika.itb.stei.itb.ac.id/~rinaldi.munir/



Gambar 2.3 Contoh-contoh graf semu
Sumber : https://informatika.stei.tib.ac.id/~rinaldi.munir/

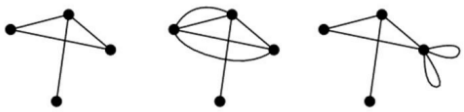
Selain berdasarkan kesederhanaan graf, graf juga bisa dibagi berdasarkan orientasi arah pada sisi. Terdapat 2 jenis graf berdasarkan kategori tersebut yaitu graf berarah dan graf tak berarah.

1. Graf Berarah (*Directed Graph*)
   Graf berarah adalah graf yang setiap sisinya diberikan orientasi arah. Perbedaannya dengan graf tak berarah biasanya bisa dilihat dari panah yang merepresentasikan arah sisi. Beginilah contoh-contoh dari graf berarah.



Gambar 2.4 Contoh-contoh graf berarah
Sumber : https://informatika.stei.tib.ac.id/~rinaldi.munir/

2. Graf Tak Berarah (*Undirected Graph*)
   Graf tak berarah adalah graf yang sisinya tidak mempunyai orientasi arah. Biasanya graf tak berarah tidak mempunyai panah karena graf-nya tidak memiliki arah. Beginilah contoh-contoh dari graf tak berarah.



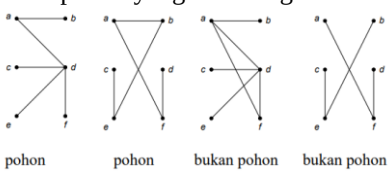
Gambar 2.5 Contoh-contoh graf tak berarah
Sumber : https://informatika.stei.tib.ac.id/~rinaldi.munir/

Pada graf, terdapat beberapa terminologi yang penting dalam mempelajari sebuah graf. Berikut adalah terminologi-terminologi dalam sebuah graf.

1. Ketetanggaan (*Adjacent*)
   Misalkan terdapat simpul A dan simpul B. Jika sebuah sisi menghubungkan A dengan B, A dan B bisa dikatakan bertetangga karena keduanya terhubung langsung.
2. Bersisian (*Incidency*)
   Misalkan terdapat sebuah simpul A dan B. Jika sisi sembarang menghubungkan simpul A dan B tersebut, sisi tersebut beririsan dengan simpul A dan B.
3. Simpul Terpencil (*Isolated Vertex*)
   Simpul terpencil adalah simpul yang tidak mempunyai sisi yang bersisian dengan simpul lain. Ini artinya simpul tersebut tidak berhubungan dengan simpul apapun sehingga tersendiri.
4. Graf Kosong (*Null Graph*)
   Sebuah graf disebut kosong jika semua simpul tidak mempunyai sisi yang menghubungkan simpul tersebut ke simpul yang lain. Ini artinya semua simpul yang ada graf tersebut adalah simpul terpencil.
5. Derajat (*Degree*)
   Derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut. Misalkan sebuah simpul A mempunyai sisi yang menghubungkan A dengan B,C, dan D. Ini artinya simpul A memiliki derajat sebesar 3. Derajat bisa direpresentasikan dalam bentuk d(x). Jika simpul A memiliki derajat 3, bentuk notasinya adalah d(3).
6. Lintasan (*Path*)
   Dari kumpulan-kumpulan sisi yang ada pada sebuah graf, lintasan bisa dibuat dari simpul awal v_0 ke simpul tujuan v_n selama dari simpul ke simpul lain selalu terdapat sisi yang menghubungkan kedua simpul tersebut.
7. Siklus (*Cycle*) atau Sirkuit (*Circuit*)
   Sirkuit atau siklus adalah lintasan yang berawal dan berakhir pada simpul yang sama
8. Keterhubungan (*Connected*)
   Dua buah simpul v_1 dan simpul v_2 disebut terhubung jika terdapat lintasan dari v_1 ke v_2. Graf disebut graf terhubung jika semua simpul memiliki lintasan dengan simpul lainnya. Jika tidak, maka graf tersebut disebut sebagai graf tak-terhubung.
9. Upagraf (*Subgraph*)
   Misalkan graf A yang simpul dan sisinya merupakan bagian dari graf B, maka graf A merupakan upagraf dari graf B. Graf B yang bukan merupakan bagian dari graf A adalah komplemen dari sebuah upagraf.

B. Pohon (Tree)

Pohon adalah graf. Tetapi graf dari pohon tidak mengandung sirkuit, tidak memiliki arah, dan selalu terhubung. Pohon juga bisa didefinisikan sebagai struktur data yang terdiri dari simpul yang memiliki anak pohon yang dihubungkan oleh sisi-sisi.



Gambar 2.6 Contoh-contoh graf pohon dan bukan pohon
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/

Pohon memiliki sejumlah sifat-sifat atau properti. Misalkan G = (V,E) adalah sebuah graf tak-berarah sederhana dan jumlah simpulnya n. Beginilah sifat-sifat graf tersebut

1. G adalah pohon
2. Setiap pasang simpul di G terhubung dengan lintasan tunggal
3. G terhubung dan sisinya berjumlah n-1
4. G tidak ada sirkuit
5. Penambahan satu sisi pada graf akan membuat satu sirkuit
6. Semua sisi G adalah jembatan

Pohon yang digunakan dalam makalah ini adalah *K-D Tree*. Jenis pohon ini adalah pohon berakar. Berikut ini adalah terminologi dari pohon berakar.

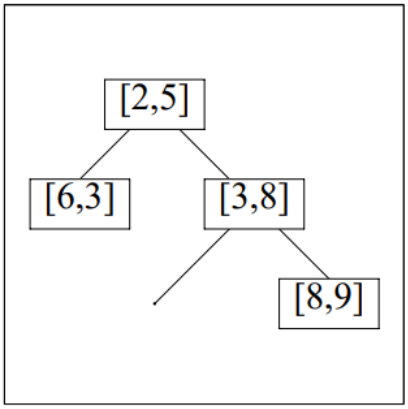
1. Anak (*children*) dan Orangtua (*parent*)
   Misalkan simpul A berhubungan dengan simpul B,C, dan D dan simpul B,C, dan D tidak berhubungan satu sama lain, maka A adalah orangtua dari B,C,D dan B,C,D adalah anak-anak dari A.
2. Lintasan (*path*)
   Pada pohon, lintasan akan selalu ada dari simpul manapun karena pohon merupakan graf terhubung. Untuk menghitung panjang lintasan, hitung sisi-sisi dari simpul awal ke simpul tujuan.
3. Saudara kandung (*sibling*)
   Simpul-simpul yang memiliki orangtua yang sama adalah saudara kandung
4. Upapohon (*subtree*)
   Misalkan terdapat pohon yang memiliki anak pohon. Maka anak pohon tersebut adalah upapohon dari pohon awal tersebut.
5. Derajat (*degree*)
   Derajat adalah jumlah upapohon atau jumlah anak yang dimiliki oleh sebuah simpul. Jika simpul memiliki 4 anak, maka derajatnya adalah 4.
6. Daun (*leaf*)
   Simpul yang tidak memiliki cabang atau anak disebut sebagai daun.
7. Simpul Dalam (*internal nodes*)
   Sebaliknya, simpul yang memiliki adak adalah simpul dalam
8. Tingkat (*level*)
   Tingkat merepresentasikan ketinggian sebuah pohon. Misalkan simpul A tingkatnya 0, maka anak dari A tingkatnya 1 dan seterusnya.
9. Tinggi (*height*) atau Kedalaman
   Tingkat maksimum dari pohon disebut tinggi atau kedalaman pohon.

C. Pohon K-D (K-D Tree)

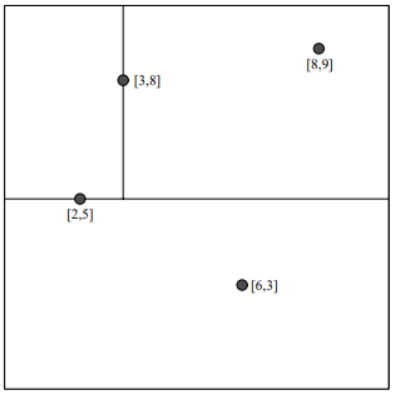
Pohon K-D adalah sebuah struktur data yang berguna untuk menyimpan sekumpulan titik tertentu dari ruang berdimensi k. Sebuah pohon k-d adalah pohon biner. Isi dari setiap simpul adalah sebagai berikut

1. Vektor domain
2. Vektor range
3. Pemisah dimensi
4. Pohon kiri
5. Pohon kanan

Beginilah contoh dari sebuah pohon k-d.



Gambar 2.7 Pohon 2 dimensi dengan 4 elemen
Sumber :
https://www.ri.cmu.edu/pub_files/pub1/moore_andrew_1991_1/moore_andrew_1991_1.pdf



Gambar 2.8 Pohon 2 dimensi dalam bidang xy
Sumber :
https://www.ri.cmu.edu/pub_files/pub1/moore_andrew_1991_1/moore_andrew_1991_1.pdf

D. Minutiae

Dalam sebuah sidik jari, titik minutiae adalah fitur utamanya. Keunikan dari sebuah sidik jari bisa ditentukan keunikannya dengan menggunakan titik minutiae tersebut. 25-80 minutiae adalah standar untuk gambar sidik jari yang berkualitas. Kualitas dari gambar sidik jari bisa tergantung kepada *scanner*, peletakkan jari, dll.

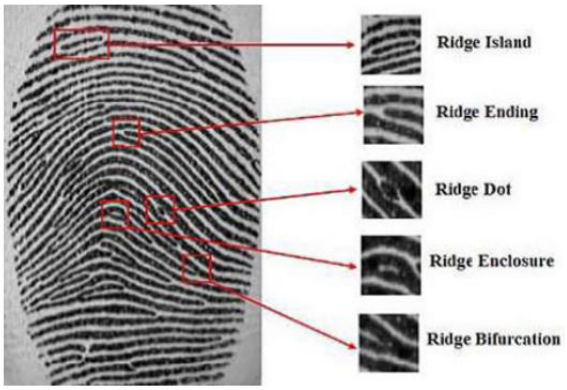
Minutiae sendiri adalah representasi dari titik dari garis punggungan yang bercabang atau berakhir. Titik-titik dimana punggungan berakhir bermacam-macam. Berikut adalah jenis-jenisnya

1. *Ridge Ending*
   Titik yang punggungannya berakhir
2. *Ridge Bifurcation*
   Titik yang punggungannya bercabang menjadi 2 atau lebih punggungan
3. *Ridge Dots*
   Punggungan yang sangat kecil
4. *Ridge Islands*

Punggungan yang mengambil area tengah diantara dua pungunggan yang bercabang

5. *Ponds or Lakes*
   Area kosong dari punggungan yang bercabang
6. *Spurs*
   Takik yang menonjol dari punggung bukit
7. *Bridges*
   Punggungan kecil yang menyatukan 2 punggungan
8. *Crossovers*
   Crossover dibuat ketika dua punggungan menyebrang satu sama lain

Dari semua jenis punggungan yang ada. Punggungan yang paling umum digunakan adalah *ridge endings* dan *ridge bifurcation* karena kedua jenis punggungan ini adalah jenis yang paling mendasar. Beginilah contoh punggungan dari sebuah sidik jari.



Gambar 2.9 Punggungan minutiae pada sidik jari
Sumber : https://www.bayometric.com/minutiae-based-extraction-fingerprint-recognition/

III. ANALISIS

A. Metode Analisis

Metode yang digunakan untuk mendapatkan solusi bagaimana caranya mengimplementasikan K-D Tree dan Minutiae untuk mengidentifikasi sidik jari adalah dengan melakukan studi pustaka dan membuat simulasi identifikasi sidik jari dengan menggunakan bahasa pemrograman Python.

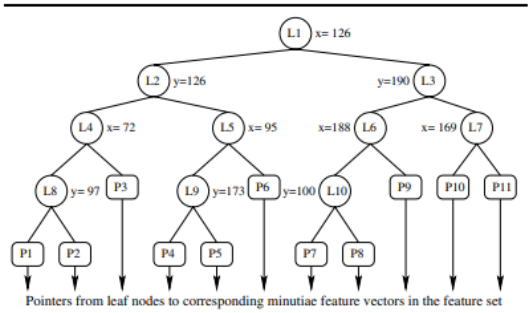
B. Minutiae Extraction and K-D Tree Storing

Pada metode ini, sebuah simpul minutiae direpresentasikan sebagai

$$p_i = (x_i, y_i, t_i, \varphi_i, \lambda_i, s_i) \quad (1)$$

Di persamaan ini, x_i, y_i merepresentasikan koordinat bidang xy dari p_i, t_i adalah tipe dari minutiae yang terdiri dari *ridge termination* dan *ridge bifurcation*, φ_i adalah arah dari punggungan local pada p_i, λ_i adalah panjang dari punggungan lokal p_i, dan s_i merepresentasikan nilai yang berhubungan dengan p_i dan derajat keaslian dari p_i.

Metode identifikasi sidik jari dilakukan dengan membandingkan sebuah gambar sidik jari dengan sebuah dataset yang berisi berbagai sidik jari. Gambar sidik jari yang akan diidentifikasi dikonversi menjadi nilai p_i dan kumpulan p_i akan diproses ke dalam suatu set yang berupa S_i. Tujuan dari ekstraksi ini adalah untuk membandingkan S_i dan S_k untuk dilihat seberapa besar kesamaan dari kedua *set* tersebut. Untuk membandingkannya, posisi koordinat akan disimpan di dalam K-D Tree. Beginilah contoh dari sebuah K-D Tree yang menyimpan simpul minutiae.



Gambar 3.1 K-D Tree yang menyimpan minutiae
Sumber : Approximate fingerprint matching using kd-tree

K-D Tree ini menyimpan absis dan ordinat dari minutiae. Dalam makalah ini, jumlah dimensi yang digunakan untuk mengidentifikasi sidik jari adalah 2. Kompleksitas algoritma untuk melakukan pengecekan adalah $O(\sqrt{n} + m)$. Simbol n melambangkan jumlah simpul yang disimpan dan m melambangkan jumlah kecocokan yang ada di pertanyaan tersebut.

C. Penjelasan Program

K-D Tree yang menyimpan simpul minutiae tersebut akan diproses untuk dicari kecocokannya. Oleh karena itu, untuk melakukan penyocokan tersebut, penulis membuat program dengan bahasa pemrograman Python untuk memudahkan proses penyocokan. Beginilah struktur data Node sebagai simpul dan garis dari sebuah K-D Tree yang direpresentasikan dalam bentuk objek.

```
class Node:
    def __init__(self,points,left,right):
        self.points = points
        if left == None:
            self.left = None
        else:
            self.left = left
        if right == None:
            self.right = None
        else:
            self.right = right
```

Karena dimensinya adalah 2, komponen yang berupa points ada dua tipe yaitu absis (x) dan ordinat(y). Adapun komponen left melambangkan anak kiri dari root dan komponen kanan melambangkan anak kanan dari root. Ini adalah struktur data K-D Tree dengan menggunakan Class.

```
class KDTree:
    def __init__ (self, minutiae,d=0):
        if not minutiae:
            self.root = None
        else:
            length = len(minutiae[0])
            axis = d % length
            s_points =
sorted(minutiae,key=lambda x: x[axis])
```

```
            med_id = len(s_points) // 2
            self.root =
Node(s_points[med_id],None,None)
            self.root.left =
KDTree(s_points[:med_id], d+1)
            self.root.right =
KDTree(s_points[med_id+1:],d+1)
```

Seperti yang sudah dibahas sebelumnya, untuk mencari kesamaan dari sebuah sidik jari, perlu dilakukan sebuah ekstraksi minutiae. Dengan library yang bernama OpenCV, hal ini bisa dilakukan tetapi dengan metode yang berbeda. Beginilah program yang dibuat untuk melakukan ekstraksi minutiae.

```
from cv2 import cv2
import numpy as py

def extract_minutiae(image_path):
    image = cv2.imread(image_path,
            cv2.IMREAD_GRAYSCALE)
    image_blur = cv2.GaussianBlur(image,(5,5),0)
    _,image_thresh = cv2.threshold(image_blur,
                     120,255,cv2.THRESH_BINARY_INV)
    image_contours,_ = cv2.findContours
                       (image_thresh,
                       cv2.RETR_EXTERNAL,
                       cv2.CHAIN_APPROX_SIMPLE)
    image_minutiae = np.zeros_like(image,
                     dtype=np.uint8)
    far_points = []
    for i in image_contours:
        hull = cv2.convexHull(i,returnPoints=False)
        try:
            defects = cv2.convexityDefects(i,
                      np.array([hull]))
            if defects is not None:
                for j in range(defects.shape[0]):
                    _, _, f, _ = defects[j, 0]
                    far = tuple(i[f][0])
                    far_points.append(far)
                    cv2.circle(image_minutiae, far,
                               3, (255, 255, 255),
                               -1)
        except cv2.error as e:
            continue
    return far_points
```

Beginilah tahap-tahapan ekstraksi minutiae dari sebuah gambar.

1. Program membaca sebuah gambar dengan menggunakan library OpenCV. Gambar dikonversi menjadi skala abu-abu
2. Untuk mengurangi *noise*, program membuat gambar yang diproses menjadi buram
3. Dengan menggunakan cv2.threshold, program menentukan ambang atau *threshold* dari gambar.
4. Program melakukan deteksi akan kontur dengan menggunakan cv2.findContours
5. Setiap kontur yang ada pada gambar, program melakukan ekstraksi akan titik-titik minutiae yang ada
6. Hull konveks dari setiap kontur dihitung dan pada hull konveks tersebut ditentukanlah cacat-cacat yang ada
7. Pada setiap cacat yang didapat, program mencari titik jauh pada program dan dimasukan ke dalam sebuah array untuk diproses.
8. Proses diulang sampai semua kontur pada gambar sudah diproses. Setelah semuanya sudah diproses, program mengembalikan sebuah array yang berisi titik-titik minutiae
9. Titik-titik minutiae akan diproses pada program selanjutnya

Aplikasi pohon K-D akan digunakan pada pencocokan sidik jari. Program menerima titik-titik minutiae input gambar, pohon K-D yang berisi titik-titik minutiae, dan ambang yang ingin kita masukkan. Tidak seperti cara yang ada pada bagian B, program tidak memproses 2 K-D Tree melainkan hanya memproses 1 array dan 1 K-D Tree.

```
def match_fingerprints(q_point, r_tree,
                       threshold):
    matches = []
    for i in q_point:
        n_point = r_tree.find_
                  nearest_neighbor(i)
        dis = euclidean_
              distance(i,n_point)
        if dis <= threshold:
            matches.append((i,n_point))
    return matches
```

K-D Tree ini digunakan untuk mencari tetangga terdekat seperti pada program ini.

```
def find_nearest_neighbor(self,q_point):
        best = None
        best_dist = float('inf')

        def find_nearest_neighbor
        _helper(node, depth=0):
            nonlocal best, best_dist
            if node is None:
                return
            k =  len(q_point)
            axis = depth % k
            closer_subtree = None
            farther_subtree = None
            if q_point[axis] <
               node.points[axis]:
                closer_subtree =
                node.left
                farther_subtree =
                node.right
            else:
                closer_subtree =
                node.right
                farther_subtree =
                node.left
            find_nearest_neighbor_helper
            (closer_subtree.root,depth+1)
            dist = sum((q_point[i]
                   node.points[i]) ** 2
                   for i in range(k))
            if dist < best_dist:
                best = node.points
                best_dist = dist
            if abs(q_point[axis]
            - node.points[axis]) <
```

```
            best_dist:
             find_nearest_neighbor_helper
             (farther_subtree.root,
             depth+1)
        find_nearest_neighbor_helper
        (self.root)
        return best
```

Berikut adalah penjelasan dari program pencarian tetangga terdekat.

1. Inisalisasi variabel terlebih dahulu karena ini akan menjadi value outputnya
2. Fungsi rekursif dibuat untuk melakukan pencarian secara rekursif karena struktur data berupa K-D Tree
3. Jumlah dimensi dari titik-titik minutiae yang ada dihitung dan dimasukkan ke variabel ‘k’. Setelah itu, sumbu dihitung untuk menentukan jumlah ruang pada K-D Tree
4. K-D Tree dibuat dengan menentukan subtree apa saja yang lebih jauh dan apa saja yang lebih dekat.
5. Fungsi bantuan yang tadi dipanggil lagi dengan parameter K-D Tree berupa subtree yang lebih dekat dari node
6. Jarak antara titik sekarang di dalam KD-Tree dan q_point.
7. Jarak yang terkecil adalah variabel ‘best’ yang merupakan ‘best_dist’ terkecil dari program
8. Lalu fungsi dipanggil lagi untuk memeriksa subtree yang lebih jauh jaraknya dari node untuk bisa melakukan perbandingan lebih kecil dengan variabel ‘best’
9. Keluaran fungsi ini adalah ‘best’

Setelah pencarian selesai, maka titik yang dioperasikan dengan titik gambar input untuk mencari jarak Euclideannya yang direpresentasikan dengan program dibawah ini.

```
def euclidean_distance(point1,point2):
    if point1 is None or point2 is None:
        return float('inf')
    return ((point1[0]-point2[0])**2 +
           (point1[1]-point2[1]) ** 2) **
            0.5
```

Jika hasil jarak yang didapat kurang dari atau sama dengan ambang, maka titik dan tetangganya akan dimasukkan ke dalam sebuah list. Dari list tersebut, akan diproses dengan program seperti ini.

```
num_matched = len(matched_points)
     similarity_score =
     (num_matched/len(minutiae_image))
     *100
     if (similarity_score > 20):
          print("File:",filename)
          print("Similarity Score:",
                similarity_score)
```

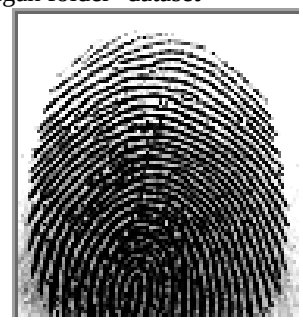
D. Hasil Program

Untuk melakukan tes, perlu ada dataset yang berisi kumpulan gambar sidik jari untuk dibandingkan dengan sidik jari lainnya. Beginilah contoh dari dataset yang penulis punya.

Gambar 3.2 Contoh dataset berisi kumpulan sidik jari
Sumber : https://www.kaggle.com/datasets/ruizgara/socofing

Untuk dataset-dataset lainnya, penulis akan melampirkan tautan pada bagian lampiran. Pada program ini, akan ditunjukkan nama file yang nilai kesamaannya lebih dari 50% dan ambangnya adalah 2.

1. test1.png dengan folder ‘dataset’



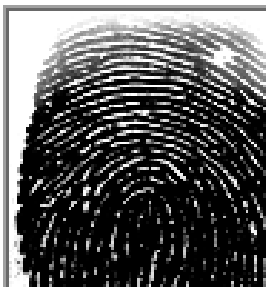
Gambar 3.3 Query sidik jari ke-1 (test1.BMP)
Sumber : https://www.kaggle.com/datasets/ruizgara/socofing

```
Enter the image path: test.BMP
Enter the dataset path: dataset
Showing the result that has
similarity score above 50%
File: 12__M_Left_middle_finger.BMP
Similarity Score: 100.0
File: 143__M_Right_ring_finger.BMP
Similarity Score: 60.0
File: 148__M_Right_index_finger.BMP
Similarity Score: 60.0
File: 151__M_Right_ring_finger.BMP
Similarity Score: 80.0
File: 153__M_Right_index_finger.BMP
Similarity Score: 60.0
File: 170__M_Left_index_finger.BMP
Similarity Score: 60.0
File: 172__M_Right_index_finger.BMP
Similarity Score: 60.0
File: 243__M_Left_thumb_finger.BMP
Similarity Score: 60.0
.
.
.
.
.
.
.
.
.
.
.
.
.
```

Gambar 3.4 Sidik jari yang memiliki kesamaan 80% dengan query (151__M_Right_ring_finger.BMP)
Sumber : https://www.kaggle.com/datasets/ruizgara/socofing

Sekilas, gambar 3.3 dan gambar 3.4 memiliki kesamaan yang cukup besar meskipun jarinya berbeda (Gambar 3.3 adalah jari tengah kiri). Oleh karena itu, kedua gambar memiliki kesamaan 80%.

2. test2.png dengan folder ‘dataset’



Gambar 3.5 Sidik jari yang akan diproses (test2.BMP)
Sumber : https://www.kaggle.com/datasets/ruizgara/socofing

```
Enter the image path: test2.BMP
Enter the dataset path: dataset
Showing the result that has
similarity score above 50%
File: 205__F_Right_index_finger.BMP
Similarity Score: 57.14285714285714
File: 256__M_Left_index_finger.BMP
Similarity Score: 57.14285714285714
File: 375__M_Right_index_finger.BMP
Similarity Score: 57.14285714285714
File: 3__M_Right_index_finger.BMP
Similarity Score: 100.0
```

Gambar 3.6 Sidik jari yang memiliki kesamaan sebasar 57,14% (M_Right_index_finger.BMP)
Sumber : https://www.kaggle.com/datasets/ruizgara/socofing

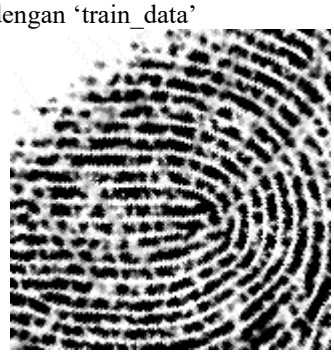
3. test3.png dengan ‘dataset’

Gambar 3.7 Sidik jari yang akan diproses (test3.png)
Sumber : Dokumen Penulis

```
Enter the image path: test3.png
Enter the dataset path: dataset
Showing the result that has
similarity score above 50%
It seems that none of the datasets
has similarity score above 50% when
compared to the query!
```

Berdasarkan pesan yang diberikan dari program, terlihat bahwa semua dataset yang ada tidak memiliki kecocokan yang lebih dari 50% dibandingkan dengan gambar ‘test3.png’. Oleh karena itu, tidak ada nama gambar yang ditunjukkan di terminal.

4. test6.BMP dengan ‘train_data’



Gambar 3.8 Sidik jari yang akan diproses (test6.BMP)
Sumber : https://www.kaggle.com/datasets/peace1019/fingerprint-dataset-for-fvc2000-db4-b

```
Enter the image path: test6.BMP
Enter the dataset path: train_data
Showing the result that has
similarity score above 50%
File: 00000_15.bmp
Similarity Score: 100.0
```

IV. KESIMPULAN

K-D Tree adalah representasi pohon biner yang simpulnya berisi angka-angka berjumlah K dengan K merepresentasikan jumlah dimensi dari simpul. Dengan adanya program dengan bahasa pemrograman Python ini, Struktur data dari K-D Tree bisa dibuat untuk melakukan proses penyocokkan satu sidik jari dengan sidik jari lainnya dengan melakukan ekstraksi minutiae juga.

Simulasi program yang dibuat ini bisa bermanfaat untuk

mencari mengidentifikasi pemilik dari sidik jari. Namun, program yang dibuat ini masih tidak cukup untuk diaplikasikan ke dunia nyata karena masih sangat sederhana. Untuk membuat penyocokkan sidik jari yang lebih efektif, perlu adanya algoritma dan pendekatan yang lebih kompleks seperti pendekatan berdasarkan studi pustaka yang dilakukan dan bisa saja dengan penggunaan *Machine Learning*.

V. UCAPAN TERIMA KASIH

Demikianlah makalah ini. Penulis ingin mengucap syukur kepada Tuhan Yang Maha Esa karena atas berkat dan anugerahnya, saya bisa menyelesaikan makalah mata kuliah IF2120 Matematika Diskrit dengan baik dan lengkap. Penulis juga ingin berterima kasih kepada semua dosen di mata kuliah Matematika Diskrit yaitu Dr. Nur Ulfa Maulidevi, S.T., M.Sc., Dr. Fariska Zakhralativa Ruskanda, S.T., M.T., Dr. Ir. Rinaldi Munir, M. T., dan Monterico Adrian, S.T., M.T. untuk bimbingannya selama 6 bulan perkuliahan ini. Penulis juga ingin mengucapkan terima kasih kepada para penulis jurnal, artikel, dll yang penulis jadikan referensi untuk membuat makalah ini. Penulis juga ingin meminta maaf sebesar-besarnya jika ada kesalahan kata yang menyinggung dalam makalah ini.

REFERENSI

[1] A. W. Moore (1990). Efficient memory-based learning for Robot Control. University of Cambridge. Computer Laboratory. Diakses pada tanggal 9 Desember 2023.
[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf. Diakses pada tanggal 8 Desember 2023.
[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/22-Pohon-Bag1-2023.pdf. Diakses pada tanggal 8 Desember 2023.
[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/23-Pohon-Bag2-2023.pdf. Diakses pada tanggal 8 Desember 2023.
[5] https://www.bayometric.com/minutiae-based-extraction-fingerprint-recognition/. Diakses pada tanggal 9 Desember 2023.
[6] https://www.kaggle.com/datasets/peace1019/fingerprint-dataset-for-fvc2000-db4-b. Diakses pada tanggal 10 Desember 2023
[7] https://www.kaggle.com/datasets/ruizgara/socofing. Diakses pada tanggal 10 Desember 2023
[8] https://xlinux.nist.gov/dads/HTML/kdimensional.html. Diakses pada 9 Desember 2023
[9] N. Kaushal, & P. Kaushal (2011). Human identification and fingerprints: A Review. Journal of Biometrics & Biostatistics, 02(04). https://doi.org/10.4172/2155-6180.1000123. Diakses pada tanggal 8 Desember 2023.
[10] P. Bhowmick, & B. B. Bhattacharya (2004). Approximate fingerprint matching using KD-tree. Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. https://doi.org/10.1109/icpr.2004.1334194. Diakses pada tanggal 9-10 Desember 2023
[11] R.A. David, Ridgeology Modern Evaluative Friction Ridge Identification. Diakses pada 8 Desember 2023.
[12] V.M., Praseetha, & S. Vadivel (2019). Enrolment and matching of fingerprints using minutiae tree. Journal of Computer Science, 15(3), 357–371. https://doi.org/10.3844/jcssp.2019.357.371. Diakses pada tanggal 9 Desember 2023

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 10 Desember 2023

Ttd (scan atau foto ttd)

Marvin Scifo Y. Hutahaean – 13522110

LAMPIRAN

Link Program :
https://github.com/scifo04/Fingerprint_KD_Tree