TUGAS AKHIR - KI141502

PENGEMBANGAN METODE *CANCELABLE TEMPLATE* PADA SIDIK JARI MENGGUNAKAN *CONVEX HULL* DAN PROYEKSI GARIS

BURHANUDIN RASYID
NRP 5113100046

Dosen Pembimbing
Tohari Ahmad, S.Kom., MIT., Ph.D.

Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

PENGEMBANGAN METODE *CANCELABLE TEMPLATE* PADA SIDIK JARI MENGGUNAKAN *CONVEX HULL* DAN PROYEKSI GARIS

BURHANUDIN RASYID
NRP 5113100046

Dosen Pembimbing
Tohari Ahmad, S.Kom., MIT., Ph.D.

Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

DEVELOPING OF CANCELABLE TEMPLATE METHOD FOR FINGERPRINT USING CONVEX HULL AND LINE PROJECTION

BURHANUDIN RASYID
NRP 5113100046

Supervisor
Tohari Ahmad, S.Kom., MIT., Ph.D.

Department of Informatics
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENGEMBANGAN METODE *CANCELABLE TEMPLATE* PADA SIDIK JARI MENGGUNAKAN *CONVEX HULL* DAN PROYEKSI GARIS

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
BURHANUDIN RASYID
NRP: 5113100046

Disetujui oleh Pembimbing Tugas Akhir:

1. Tohari Ahmad, S.Kom., MIT., Ph.D. .....
   (NIP. 197505252003121002) (Pembimbing)

SURABAYA
MEI, 2017

(Halaman ini sengaja dikosongkan)

PENGEMBANGAN METODE *CANCELABLE TEMPLATE* PADA SIDIK JARI MENGGUNAKAN *CONVEX HULL* DAN PROYEKSI GARIS

Nama Mahasiswa	**: BURHANUDIN RASYID**
NRP	**: 5113100046**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing	**: Tohari Ahmad, S.Kom., MIT,. Ph.D.**

Abstrak

Autentikasi menggunakan biometrik khususnya sidik jari sudah sering digunakan karena kemudahannya. Salah satu metode autentikasi sidik jari yaitu Cancelable Template yang merupakan metode dalam autentikasi sidik jari yang aman dan bersifat irreversible. Sehingga pengguna tidak perlu kuatir apabila template dicuri karena tidak bisa dikembalikan ke data sidik jari semula dan dapat membuat template dengan key yang baru lagi. Namun untuk mendapatkan keamanan yang kuat, waktu proses yang cepat, serta nilai akurasi yang tinggi sulit diperoleh.

Tugas akhir ini mengusulkan pengembangan dari metode Cancelable Template menggunakan convex hull pada proses seleksi titik dan proyeksi garis untuk menentukan representasi data. Seleksi titik yang tepat dengan area threshold tertentu mempercepat metode yang diusulkan. Selain itu penambahan jumlah sektor, penambahan tahapan transformasi, dan penambahan variable pada representasi data membuat metode ini aman.

Hasil uji coba terhadap dataset menghasilkan EER sebesar 5,69 % lebih akurat dari metode sebelumnya dengan selisih 3,69%. Kecepatan proses verifikasi menjadi 5 x lipat lebih cepat dibandingkan dengan tanpa adanya seleksi titik serta keamanan dengan 5,184 x 10^{16} kemungkinan dapat kembali ke data asal.

Kata kunci: Cancelable Template, Convex Hull, Proyeksi Garis, Verifikasi.

(Halaman ini sengaja dikosongkan)

DEVELOPING OF CANCELABLE TEMPLATE METHOD FOR FINGERPRINT USING CONVEX HULL AND LINE PROJECTION

Student's Name	**: BURHANUDIN RASYID**
Student's ID	**: 5113100046**
Department	**: Teknik Informatika FTIF-ITS**
Supervisor	**: Tohari Ahmad, S.Kom., MIT,. Ph.D.**

Abstract

Authentication using biometrics fingerprint in particular have often used because its convenience. One method of fingerprint authentication is Cancelable Template which is a method of fingerprint authentication with safe and irreversible. So users don't have to worry if the template is stolen because it can not be restored to the original fingerprint data and can create a template with the new key again. But to get a strong security, fast processing time, as well as high accuracy values are difficult to otbtain.

This final project propose the development of cancelable template method using convex hull on the process of point selection and line projection to determine the data representation. Selection of the right with a specific threshold area speeds up the proposed method. Beside that, the addition of the number of sectors, the addition of transformation stages, and the addition of variables in the data representation make this methode safe.

The trial result of the data produces the EER of 5.69% more accurate than the previous method with 3.69% difference. The speed of the verification process is five times faster than with no point selection and security with 5.184 x 10^{16} possibilites can be returned to the original date.

Keywords : Cancelable Template, Convex Hull, Line Projection, Verification.

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, karena limpahan rahmat dari Allah Swt sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**PENGEMBAGNAN METODE *CANCELABLE* TEMPLATE PADA SIDIK JARI MENGGUNAKAN *CONVEX HULL* DAN PROYEKSI GARIS**"

sebagai salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Segala puji bagi Allah Subhanahu Wa Ta'ala dan shalawat serta salam kepada junjungan Nabi Muhammad Sallallahu Alaihi Wasallam.
2. Keluarga penulis (Bapak, Ibu, Kakak, dan Adik) yang telah memberikan dukungan moral, spiritual dan material serta senantiasa memberikan doa demi kelancaran dan kemudahan penulis dalam mengerjakan Tugas Akhir.
3. Bapak Tohari Ahmad, S.Kom., MIT,. Ph.D. selaku pembimbing yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Bapak Dr. Eng. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator mata kuliah Tugas Akhir yang memberikan banyak bantuan informasi.
5. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
6. Admin LAB RMK KBJ yang telah banyak membantu memfasilitasi sarana dan prasarana dalam pengerjaan.

7. Teman-teman RMK KBJ yang telah berjuang bersama penulis.
8. Teman-teman Kontrakan Muslimin dimana tempat penulis tinggal selama 4 tahun yang menjadi keluarga di Surabaya.
9. Pihak-pihak lain yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Mei 2017

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Pada bab ini akan dibahas hal-hal yang mendasari tugas akhir. Bahasan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir.

1.1 Latar Belakang

Teknologi berkembang sangat pesat membuat manusia sangat bergantung kepada teknologi. Perkembangan teknologi ini juga selaras dengan berkembangnya keamanan dan privasi yang merupakan hal penting dalam autentikasi penggunaan teknologi. Keamanan dalam bentuk autentikasi bersifat dasar yang banyak digunakan dalam setiap aktifitas atau pekerjaan. Seperti yang sering digunakan adalah mekanisme kata sandi (*password*) yang digunakan untuk kebanyakan *website*. Lalu sistem *token* untuk beberapa aplikasi mobile. Serta yang mulai berkembang adalah menggunakan biometrik. Biometrik ada beberapa jenis seperti wajah, suara, sidik jari, dan iris[1]. Dibandingkan dengan metode autentikasi *token & password,* biometrik mempunyai banyak keuntungan. Salah satunya adalah manusia agak kesulitan mengingat *password* dengan bagus, apalagi penggantian yang berkala dari *password* itu sendiri. Sehingga perlu adanya *reset password* yang memakan waktu yang lama.

Biometrik menggunakan sidik jari menjadi sangat populer dikarenakan sifat sidik jari sendiri mempunyai sifat unik untuk masing-masing pengguna. Selain itu sidik jari lebih populer digunakan dalam sistem biometrik dari pada jenis biometrik lainnya[2]. Sidik jari mempunyai hasil evaluasi yang bagus, relatif permanen, dan mempunyai *key factor* seperti *acceptability, universality, performance* dan *collectivity*[3]. Namun, ada beberapa hal yang perlu diperhatikan dalam menggunakan sistem biometrik pada sidik jari ini. Pertama berkaitan dengan keamanan

data sidik jari, dimana data informasi sidik jari tidak boleh disimpan langsung, dikarenakan apabila data sidik jari asli dicuri oleh orang yang tidak bertanggung jawab akan berbahaya. Sehingga yang disimpan yaitu data sidik jari palsu setelah dilakukan proses modifikasi. Kedua yaitu labilnya data sidik jari, pengguna sering berubah-ubah dalam memposisikan jarinya ketika menggunakan alat sidik jari. Selain itu, alat ekstraksi sidik jari yang terkadang kurang valid mengganggu proses verifikasi sidik jari. Keamanan sidik jari ini tidak dapat diamankan menggunakan kriptografi tradisional seperti DES, AES, ataupun dilakukan dengan *hash*[4]. Perlu metode baru untuk mengamankan sidik jari sebagai autentikasi biometrik.

Salah satu metode untuk mengamankan data sidik jari yaitu metode c*ancelable template*[5]. Metode *cancelable template* adalah metode autentikasi sidik jari yang bersifat *irreversible* dimulai dari penyimpanan data sidik jari yang sudah dilakukan transformasi dengan *key* tertentu yang disebut *template* ke dalam *database*. Selanjutnya dalam autentikasi berlangsung, hasil *scanning* pengguna akan diekstraksi dan ditransformasi dengan *key* yang sama dalam pembuatan *template* yang disebut *query*. Lalu dilakukan proses pencocokan atau verifikasi antara *query* dan *template* yang ada di *database*. Karena terdapat ketidakstabilan hasil ekstraksi pada waktu *scanning* perlu adanya toleransi ketidaksamaan dan menggunakan *threshold* tertentu. Kelabilan data dipengaruhi oleh pemilihan titik *minutiae* yang salah sebagai titik pusat sehingga terjadi pencocokan palsu. Dalam metode yang sudah ada [6] ketika menggunakan semua titik *minutiae* maka akan lebih lama. Sudah ada metode untuk menyeleksi titik yang digunakan dalam pembuatan titik pusat untuk sebuah *template/query* [7]. Namun, hanya menggunakan area terluar dari data titik *minutiae* hasil *scanning* dalam seleksi titik terjadi pencocokan palsu akibat masih banyak titik labilnya. Selain itu, faktor keamanan yang kurang baik terjadi ketika tidak dilakukan transformasi secara kompleks.

Berdasarkan permasalahan di atas pada tugas akhir ini akan dikembangkan metode *cancelable template* yang memprioritaskan keamanan dan kecepatan dan juga pengembangan mekanisme pemilihan titik seleksi untuk titik-titik tertentu yang menggunakan *convex hull*, mekanisme transformasi, bentuk kunci, representasi data dengan menggunakan proyeksi garis, dan mekanisme verifikasi. Diharapkan akan dihasilkan suatu pengembangan metode untuk c*ancelable template* pada sisi akurasi, kecepatan, dan keamanan.

1.2 Rumusan Masalah

Tugas akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Fungsi apa saja yang dapat dikembangkan dan divariasikan pada setiap tahapan dalam metode *cancelable template* untuk proteksi sidik jari?
2. Bagaimana mekanisme untuk menyeleksi titik *minutiae* agar proses lebih cepat dan hasil tetap akurat dibandingkan tanpa seleksi titik?
3. Faktor apa saja yang dapat digunakan untuk mengoptimalkan hasil verifikasi data dengan representasi data yang tepat sehingga menjadikan hasil lebih akurat dan aman?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini memiliki batasan sebagai berikut:

1. Bahasa pemrograman yang digunakan untuk implementasi proteksi sidik jari dengan metode *cancelable template* ini adalah MATLAB.
2. Proses pengujian berupa data titik-titik *minutiae* hasil ekstraksi sidik jari menggunakan *dataset* yang sudah ada yaitu FVC2002DB2a.

3. Keluaran dari implementasi ini cocok tidaknya antara *template* dan *query*.
4. Perangkat lunak yang digunakan adalah *MATLAB R2013a* sebagai *IDE.*

1.4 Tujuan

Tujuan pengerjaan tugas akhir ini adalah :

1. Mengembangkan dan membuat variasi pada setiap tahapan pada metode *cancelable template.*
2. Mengembangkan mekanisme seleksi titik agar proses lebih cepat dan hasil tetap akurat dibandingkan dengan tanpa seleksi titik.
3. Menentukan mekanisme untuk mengoptimalkan hasil verifikasi data dalam bentuk representasi data agar lebih akurat dan aman.

1.5 Manfaat

Manfaat pengerjaan tugas akhir ini adalah :

1. Mendapatkan cara mengembangkan dan memberi variasi mekanisme untuk setiap tahapan pada metode *cancelable template*.
2. Mendapatkan mekanisme seleksi titik untuk menghindari pencocokan palsu.
3. Mendapatkan mekanisme untuk mengoptimalkan hasil verifikasi data dalam bentuk representasi data yang aman.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal

Tahap awal tugas akhir ini adalah menyusun proposal tugas akhir. Pada proposal, diajukan gagasan untuk menganalisis dan mengidentifikasi proteksi sidik jari menggunakan metode *cancelable template* dengan menggunakan *convex hull* dan proyeksi garis.

1.6.2 Studi Literatur

Tahap ini dilakukan untuk mencari informasi dan studi literatur apa saja yang dapat dijadikan referensi untuk membantu pengerjaan tugas akhir ini. Informasi didapatkan dari buku dan literatur yang berhubungan dengan metode yang digunakan. Informasi yang dicari adalah *cancelable template*, dan metode-metode *convex hull* untuk seleksi titik seperti *jarvis march*, metode transformasi dalam matematika, bentuk representasi data menggunakan proyeksi garis, serta *dataset* yang bisa digunakan untuk uji coba nantinya. Tugas akhir ini juga mengacu pada literatur jurnal karya Tohari Ahmad, Herleeyandi Markoni, Waskitho Wibisiono, Royana M. I. dengan judul "*Transforming minutiae for protecting fingerprint data"* yang diterbitkan pada tahun 2015 [1].

1.6.3 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal tugas akhir. Untuk membangun algoritma yang telah dirancang sebelumnya, maka dilakukan implementasi dengan menggunakan perangkat lunak yakni MATLAB R2013a. Implementasi dilakukan pada sistem operasi *Microsoft Windows 10 64-bit*.

1.6.4 Uji Coba dan Evaluasi

Pada tahap ini algoritma yang telah disusun diuji coba dengan menggunakan data uji coba yang ada. Data uji coba tersebut diujicoba dengan menggunakan perangkat lunak dengan tujuan mengetahui kemampuan metode yang digunakan dan mengevaluasi hasil tugas akhir dengan jurnal pendukung yang ada. Hasil evaluasi mencakup keamanan dari evaluasi perhitungan kemungkinan cara untuk membobol template sidik jari, kecepatan pembuatan *template/query* dan verifikasi, serta keakuratan dari hasil modifikasi akan dievaluasi dengan tingkat akurasi berupa perhitungan *Equal Error Rate* (EER) dengan menggunakan data FVC2002DB2a.

1.6.5 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencangkup metode, dasar teori, implementasi, hasil uji coba dan hasil evaluasi yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan tugas akhir ini.
3. Bab III. Perancangan Perangkat Lunak

Bab ini berisi pembahasan mengenai perancangan dari metode *cancelable template* pada sidik jari yang akan diimplementasikan dan dilakukan pengujian.

4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi modifikasi metode *cancleable template* pada sidik jari dalam bentuk *Pseudocode*.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dari modifikasi metode *cancelable template* pada sidik jari yang sudah diimplementasikan pada *Pseudocode*. Uji coba dilakukan dengan menggunakan *dataset* FVC2002DB2a. Hasil evaluasi mencakup akurasi proses autentikasi yang dihasilkan serta keamanan dari *template* sidik jari yang dihasilkan.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan tugas akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.

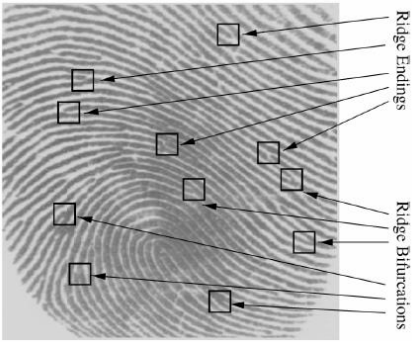
(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

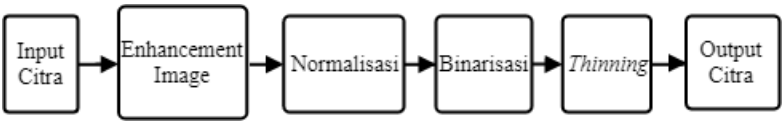
Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam tugas akhir. Teori-teori tersebut diantaranya adalah metode *cancelable template* yang didalamnya terdapat *convex hull* dalam menyeleksi titik, beberapa metode transformasi, dan beberapa teori lain yang mendukung pembuatan tugas akhir ini.

2.1 Sidik Jari

Sidik jari adalah hasil dari regenerasi epidermis ujung jari yang membentuk area lembah dan pegunungan[3]. Sidik jari merupakan salah satu jenis biometrik yang banyak digunakan saat ini. Sidik jari bersifat unik antara masing-masing individu. Hasil *scanning* sidik jari menggunakan alat *live scan* memberikan beberapa data berupa titik yang sering disebut dengan titik *minutiae.* Titik *minutiae* ini dikelompokkan menjadi dua tipe dasar pada **Gambar 2.1** yaitu *ridge ending* dan *ridge bifurcation.* Fitur dasar titik *minutiae* yang sering digunakan untuk proses verifikasi adalah koordinat posisi titik *minutiae* (*x,y*), orientasi dari titik *minutiae* (φ), serta tipe titik *minutiae* (*t*) [2].

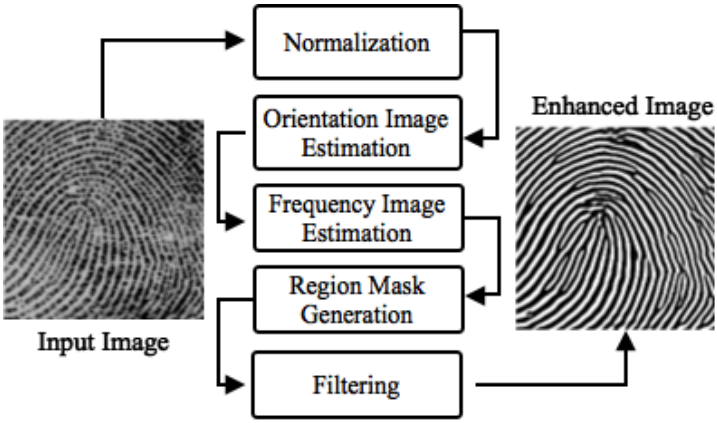


Gambar 2.1 Tipe Sidik Jari [8]



Gambar 2.2 Alur Metodelogi Ekstraksi *Minutiae*

Namun sebelumnya dilakukan ekstraksi dari gambar yang didapatkan dari alat *live scan*. Secara umum metodologi ekstraksi *minutiae* dapat dilihat pada **Gambar 2.2.** Diawali dengan input citra dijadikan citra *gray-scale* kemudian dikonversikan menjadi matriks. Selanjutnya dilakukan *enhancement image* dengan *enhancement algorithm*[8]. Pada **Gambar 2.3** merupakan langkah-langkah *enhancement image* dimulai dari *normalization*, *orientation image estimation*, *frequency image estimation*, *region mask generation*, dan *filtering*. *Enchancement algorithm* menggunakan filter Gabor menghasilkan *enhanced image* yang jelas dan kontras.



Gambar 2.3 Alur Enhancement Algorithm[8]

Selanjutnya dilakukan normalisasi lagi untuk membuat nilai intensitas dari citra standar. Kemudian *binarisasi* dapat meningkatkan kekontrasan untuk membedakan antara bukit dan lembah. *Thinning* merupakan cara untuk mengikis piksel menjadi

satu piksel lebarnya agar mudah menemukan piksel ke berapa titik *minutiae* tersebut berada. Lalu dilakukan penandaan *ridge ending* dan *ridge bifurcation* pada piksel sesuai tipe titik *minutiae*. Terakhir dilakukan penghitungan nilai x,y dan φ untuk disimpan. Jumlah titik *minutiae* dapat diekstraksi dengan kualitas baik jika terdapat 20-70 titik *minutiae*. Bisa jadi terdapat 100 lebih titik *minutiae* tergantung pengambilan sidik jari, kualitas alat *live scan*, dan metode ekstraksi titik *minutiae* yang digunakan.

Pada tugas akhir ini data sidik jari yang digunakan tidak diperoleh dari alat *live scan* secara langsung, tetapi menggunakan *database* FVC Internasional yang sudah ada yang dijelaskan pada **Subbab 2.10**. Sehingga tidak dilakukan ekstraksi titik *minutiae* dari citra gambar. Dalam tugas akhir ini, fitur dasar titik *minutiae* yang digunakan adalah koordinat posisi dan orientasi titik *minutiae*. Untuk tipe titik *minutiae* tidak digunakan karena kurang bervariasi dan hanya mengandung dua nilai antara satu dan nol.

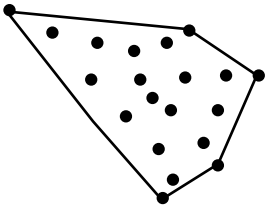
2.2 Metode *Cancelable Template*

Metode *cancelable template* beranggapan bahwa ketika sistem menyimpan data sidik jari yang selanjutnya disebut *template* dalam bentuk titik *minutiae,* maka dilakukan sebuah transformasi menggunakan *key* tertentu sehingga data yang disimpan tidak sama dengan data asli sidik jari[5]. Saat seseorang melakukan proses autentikasi, maka data hasil scanning yang selanjutnya disebut *query* akan dilakukan transformasi dengan *key* yang sama dengan *template*. Setelah itu data *template* dan *query* akan dibandingkan dengan melakukan verifikasi[9]. Dalam melakukan transformasi diberikan sebuah fungsi yang bersifat *irreversible*. Fungsi *irreversible* yaitu fungsi yang tidak dapat mengembalikan data ke bentuk semula dari data aslinya. Hal ini bertujuan apabila *template* dicuri. Pencuri tidak bisa merekonstruksi ulang ke data sidik jari aslinya, sehingga data sidik jari tidak dapat disalahgunakan.

Dalam tugas akhir ini, metode *cancelable template* menggunakan pair-polar coordinate-based[6] dan untuk meningkatkan keamanan menggunakan transformasi[7]. Umumnya terdiri dari seleksi titik, transformasi, pembuatan template dalam bentuk representasi data, dan verifikasi antar *template* dan *query*. Konsepnya hampir sama namun ada pengembangan untuk mendapatkan keamanan, akurasi, dan kecepatan yang lebih baik.

2.3 Convex Hull

Convex hull adalah permasalahan untuk membentuk suatu curva yang mampu mengcover semua himpunan titik dengan seminimal mungkin jumlah garis yang dibuat. Garis yang saling bersambung ini membentuk poligon. Poligon bisa dikatakan konveks yang dapat dilihat pada **Gambar 2.4** jika himpunan titik yang ada masuk dalam bidang poligon itu sendiri. *Convex hull* merupakan masalah dalam geometri komputasional[10].



Gambar 2.4 Polygon konveks

Terdapat beberapa algoritma untuk menyelesaikan permasalahan *convex hull.* Algoritma tersebut seperti *Bruto force, divide-and-conquer, jarvis march, quick hull, chan's algorithm,* dan algoritma lainnya. Dalam tugas akhir ini, penulis menggunakan *convex hull* pada tahap seleksi titik untuk menyeleksi titik yang digunakan sebagai pusat pembuatan *template*. Algoritma *convex hull* yang digunakan adalah *jarvis march* yang dijelaskan pada **Subbab 2.4**.

2.4 *Jarvis March*

Jarvis March merupakan suatu algoritma yang digunakan untuk menyelesaikan permasalahan *convex hull* [11]. Teori ini dipublikasikan tahun 1973 oleh R.A. Jarvis. Untuk total waktu yang digunakan oleh algoritma Jarvis March ini adalah O(nh).

Adapun langkah-langkah algoritma *jarvis march* [12] sebagai berikut:

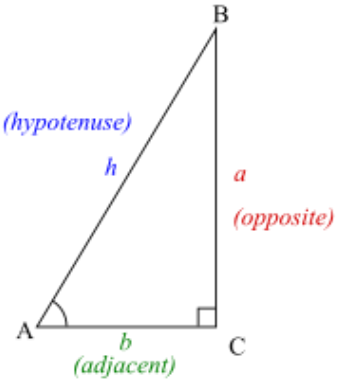
1) Tentukan titik *current* pertama yaitu titik dengan posisi titik *y* yang nilainya paling kecil, jika misalnya ada dua atau lebih titik yang memiliki nilai *y* terkecil maka pilihlah titik yang memiliki nilai *x* yang paling kecil. Atau dengan kata lain pilihlah titik yang paling pojok kiri. Masukkan titik *current* dalam hasil output.
2) Selanjutnya *update* titik *current* hingga titik *current* yang dipilih adalah titik *current* yang pertama kali dipilih. Titik-titik yang dipilih sebagai titik *current* tersebut merupakan hasil *output* dari algoritma *jarvis march*. Untuk menentukan titik *current* selanjutnya caranya yaitu dengan menghitung sudut *polar* semua titik terhadap titik *current* saat ini, titik yang memiliki sudut *polar* terkecil terhadap titik *current* saat ini adalah titik yang akan menjadi titik *current* selanjutnya. Jika misalnya terdapat 2 atau lebih titik yang memiliki sudut *polar* yang sama maka titik *current* selanjutnya adalah titik yang memiliki jarak paling jauh dari titik *current* saat ini.
3) Lakukan langkah 2 hingga sampai di titik yang memiliki y tertinggi. Jika telah sampai pada titik yang tertinggi maka yang harus dilakukan adalah mengubah *x* menjadi *-x* untuk menghitung sudut *polar*-nya. Pilih titik yang memiliki sudut *polar* terkecil untuk dipilih sebagai *current* sama seperti langkah 2.
4) Jika *current* selanjutnya ternyata adalah *current* yang dipilih pertama kali maka proses selesai.

2.5 Koordinat Polar

Sistem koordinat *polar* atau kutub adalah suatu sistem koordinat dua dimensi yang setiap titik pada bidang tertentu ditentukan dengan jarak dari suatu titik tertentu dan sudut dari arah tertentu. Titik-titik dalam koordinat polar ada yang disebut *pole* atau "kutub" inilah titik asal dan *ray* atau "sinar" dari kutub pada arah yang ditetapkan. Untuk jarak dari suatu kutub pada koordinat polar disebut *radius* atau *radial coordinate* dan sudutnya disebut *polar angle, azimuth,* atau *angular coordinate*[13]. Sudut dalam notasi polar biasanya dinyatakan dalam derajat atau radian. *Polar angle* atau sudut *polar* dalam tugas akhir ini digunakan untuk memetakan sektor dan menghitung orientasi yang selanjutnya digunakan untuk menentukan *convex hull* menggunakan metode *jarvis march*.

2.6 Trigonometri

Trigonometri merupakan cabang ilmu matematika yang mempelajari tentang relasi antara garis dan sudut suatu segitiga serta fungsi yang berbentuk dari relasi tersebut. Fungsi trigonometri terdiri dari *cosecant, cosine, cotangent, secant, sine*, dan *tangent*[14].



Gambar 2.5 Segitiga Trigonometri

Dengan melihat **Gambar 2.5** dapat dijelaskan untuk fungsi *sine=opposite/hypotenuse, cosine = adjacent/hypotenuse, tangent = opposite/adjacent, cotangent = adjacent/opposite, secant = hypotenuse/adjacent,* dan *cosecant = hypotenuse/opposite*. Trigonometri juga mempunyai *invers* atau kebalikan dari masing-masing fungsi trigonometri. Trigonometri penting digunakan pada tugas akhir ini untuk mencari sudut yang dibentuk sebuah titik *minutiae* berdasarkan informasi jarak dan koordinat yang telah diberikan. Rumus trigonometri yang sering digunakan adalah *arctan* atau dikenal dengan sebutan *invers tangent*. *Arctan* merupakan suatu relasi untuk mendapatkan besar sudut dari dua sisi yang diketahui yang ditunjukkan dalam **Persamaan (2.1)**.

$$\alpha = \arctan\left(\frac{b}{a}\right) \tag{2.1}$$

2.7 Transformasi Geometri

Transformasi geometri merupakan cabang ilmu matematika yang merupakan proses mengubah setiap titik koordinat yang dipetakan menjadi titik koordinat lain pada bidang tertentu. Transformasi geometri direpresentasikan ke bentuk matriks yang disebut matriks transformasi. Jenis-jenis dari transformasi yang dapat dilakukan yaitu translasi, refleksi, rotasi, dan dilatasi[15]. Translasi sesuai **Persamaan (2.2)** adalah transformasi yang memindahkan setiap titik koordinat pada bidang sesuai jarak dan arah tertentu. Refleksi adalah transformasi dengan memindahkan setiap titik koordinat pada bidang dengan menggunakan sifat-sifat pencerminan pada cermin datar. Terdapat beberapa tipe refleksi dengan **Persamaan (2.3)** seperti terhadap sumbu *x*, sumbu *y*, *x* = *y*, *x* = -*y*, maupun terhadap titik(0,0). Rotasi sesuai **Persamaan(2.4)** adalah transformasi dengan memutar setiap titik koordinat pada bidang dengan menggunakan titik pusat tertentu. Dilatasi sesuai **Persamaan (2.5)** adalah transformasi dengan

melakukan pembesaran atau pengecilan bidang pada titik koordinat tertentu.

$$\mathrm{T}\begin{pmatrix}x\\y\end{pmatrix}\xrightarrow{\begin{bmatrix}a\\b\end{bmatrix}}\mathrm{T}'\begin{pmatrix}x+a\\y+b\end{pmatrix} \tag{2.2}$$

$$\begin{cases}\mathrm{T}\begin{pmatrix}x\\y\end{pmatrix}\xrightarrow{sumbu\ x}\mathrm{T}'\begin{pmatrix}x\\-y\end{pmatrix}\\ \mathrm{T}\begin{pmatrix}x\\y\end{pmatrix}\xrightarrow{sumbu\ y}\mathrm{T}'\begin{pmatrix}-x\\y\end{pmatrix}\\ \mathrm{T}\begin{pmatrix}x\\y\end{pmatrix}\xrightarrow{x=y}\mathrm{T}'\begin{pmatrix}y\\x\end{pmatrix}\\ \mathrm{T}\begin{pmatrix}x\\y\end{pmatrix}\xrightarrow{x=-y}\mathrm{T}'\begin{pmatrix}-y\\-x\end{pmatrix}\\ \mathrm{T}\begin{pmatrix}x\\y\end{pmatrix}\xrightarrow{titik(0,0)}\mathrm{T}'\begin{pmatrix}-x\\-y\end{pmatrix}\end{cases} \tag{2.3}$$

$$A\begin{pmatrix}x'\\y'\end{pmatrix}=\begin{pmatrix}\cos a & -\sin a\\ \sin a & \cos a\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} \tag{2.4}$$

$$A\begin{pmatrix}x'\\y'\end{pmatrix}=\begin{pmatrix}k & 0\\ 0 & k\end{pmatrix}\begin{pmatrix}x-a\\y-b\end{pmatrix} \tag{2.5}$$

Transformasi geometri pada tugas akhir ini, digunakan pada tahap transformasi dengan proses rotasi, refleksi, dan translasi. Selain itu, rotasi juga digunakan untuk mereposisi semua titik pada tahap transformasi pra proses sesuai arah orientasi titik pusat.

2.8 *Euclidean Distance*

Euclidean distance adalah perhitungan jarak diantara dua titik[16]. *Euclidean distance* berkaitan *Teorema Phytagoras* yang diterapkan pada satu dimensi, dua dimensi, maupun tiga dimensi. Pada tugas akhir ini *euclidean distance* yang digunakan adalah dua dimensi yang terdapat pada **Persamaan (2.6)** untuk mencari jarak terjauh antara dua titik *minutiae* pada seleksi titik. Selain itu,

euclidean distance digunakan pada untuk mencari jarak dua titik pada representasi data.

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2} \quad (2.6)$$

2.9 *Adjacency Matrix*

Adjacency matrix merupakan cara untuk merepresentasikan *node-node* dari suatu *graph* yang berdekatan dengan *node* yang lainnya. *Element* dari *adjacency matrix* selalu berpasangan yang tercerminkan oleh diagonal matriks. Contoh *adjacency matrix* dapat dilihat pada **Gambar 2.6**. *Adjacency matrix* membantu mengoptimalkan dalam membandingkan dua *node* [17].

$$\begin{pmatrix} 2 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Gambar 2.6 Contoh *Adjacency Matrix*

Adjacency matrix pada tugas akhir ini digunakan pada tahap verifikasi antara data *template* dan *query*. Tahap verifikasi terdapat dua proses yaitu verifikasi lokal dan verifikasi global. Pada masing-masing proses verifikasi terdapat *adjacency matrix* lokal dan *adjacency matrix* global.

2.10 *Dataset* FVC

FVC merupakan singkatan *Fingerprint Verification Competition*, yaitu sebuah kompetisi dalam bidang *pattern recognition* yang dikhususkan untuk pengenalan sidik jari. Pada FVC 2002 terdapat empat basis data telah dihasilkan untuk dilakukan pengujian[18]. Informasi empat basis data tersebut terdapat pada **Tabel** 2.1. Pada tugas akhir ini, *dataset* FVC yang digunakan adalah *dataset* FVC2002DB2a. Pada FVC2002DB2a

terdapat 100 pasang *dataset* fingerprint. *Dataset* ini digunakan pada saat proses analisis untuk penentuan seleksi titik dan digunakan untuk uji coba pada saat pengujian.

Tabel 2.1 Informasi Basis Data FVC 2002

Basis Data	Tipe Sensor	Scanner	Image Size	Resolution
DB1	Optical Sensor	Identix "Touch View II"	388x374	500 dpi
DB2	Optical Sensor	Biometrika "FX2000"	296x560	569 dpi
DB3	Capacitive Sensor	Precise Biometrics "100 SC"	300x300	500 dpi
DB4	SFinGe v2.51	Syntetic fngerprint	288x384	500 dpi

2.11 MATLAB

MATLAB merupakan singkatan dari *Matrix Laboratoris*. Merupakan sebuah lingkungan komputasi numerikal dan bahasa pemrograman komputer generasi keempat. MATLAB dikembangkan oleh The MathWorks. MATLAB memungkinkan pengguna untuk manipulasi matriks, membuat plot fungsi dan data, implementasi algoritma, pembuatan antarmuka penggunaan serta sebuah antarmuka dengan program dalam bahasa lainya. MATLAB telah digunakan oleh jutaan teknikan dan ilmuwan di seluruh dunia terutama pada bidang pengolahan citra, sistem kontrol, komunikasi, dan *computational finance*[19].

MATLAB digunakan secara penuh sebagai perangkat pengembang tugas akhir ini. Versi MATLAB yang digunakan adalah MATLAB R2013a. Untuk fungsi MATLAB yang digunakan adalah adalah fungsi umum MATLAB seperti manipulasi matriks, pembuatan plot, implementasi algoritma, baca tulis *file*, menghitung waktu eksekusi program, dan fungsi lainnya.

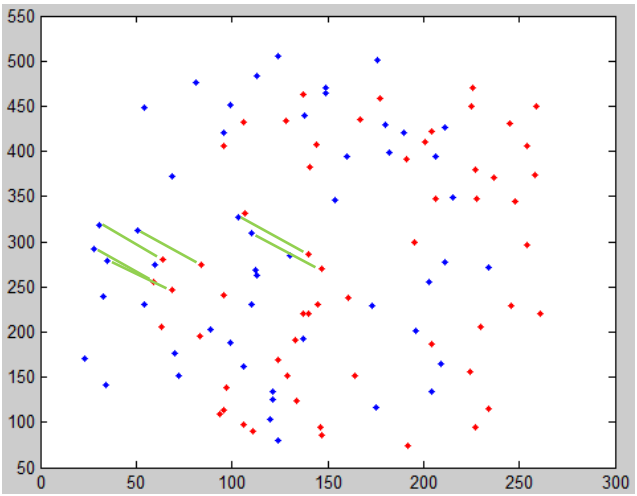
BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai perancangan dan pembuatan sistem perangkat lunak. Sistem perangkat lunak yang dibuat pada tugas akhir ini adalah pengembangan proteksi sidik jari dengan metode *cancelable template* menggunakan *convex hull* dan proyeksi garis.

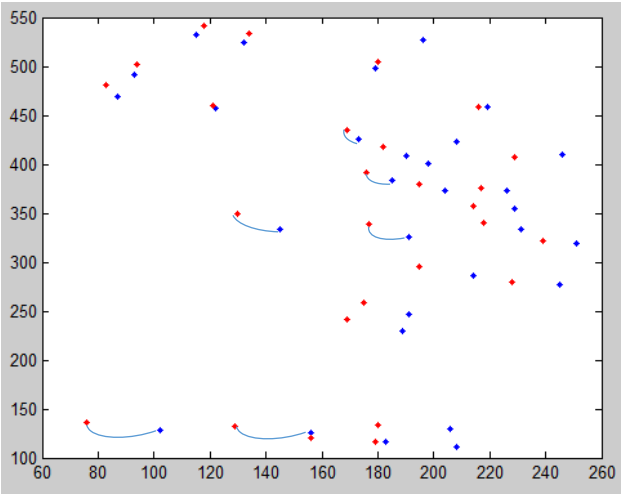
3.1 Observasi

Observasi dilakukan sebelum dilakukan implementasi dan perancangan sistem perangkat lunak. Observasi dilakukan terhadap karakteristik isi titik *minutiae* pada *dataset* FVC2002DB2a. Selain itu, observasi dilakukan dengan membuat *plotting* pada *dataset* FVC2002DB2a didapatkan perubahan posisi titik *minutiae* antara pasangan *template* dan *query* termasuk pergeseran maupun rotasi sidik jari. Dari hasil observasi yang dilakukan diketahui hasil sebagai berikut.

1. Terjadi pergeseran jari yang dilihat dari bentuk plotting dengan jari bergeser ke segala arah. Pergeseran jari dapat dilihat pada **Gambar 3.1**, dimana titik biru merupakan data *template* dan titik merah merupakan data *query*.
2. Terjadi perputaran jari yang dapat dilihat dari bentuk *plotting* searah jarum jam maupun sebaliknya. Perputaran jari dapat dilihat pada **Gambar 3.2**, dimana titik biru merupakan data *template* dan titik merah merupakan data *query*.
3. Koordinat titik *minutiae* (*x,y*) antara *template* dan *query* suka berubah-ubah karena banyak variasi nilai koordinat.
4. Besar sudut orientasi titik *minutiae (φ)* antara *template* dan *query* cukup berubah-ubah namun variasinya berbatas pada 0^o-360^o jika dirubah dari data *radian*-nya.
5. Tipe titik *minutiae (t)* sedikit berubah antara *template* dan *query* namun variasinya hanya dua yaitu nol atau satu.



Gambar 3.1 Contoh Jari yang Bergeser pada Dataset 100_1.tif.txt (biru) & 100_2.tif.txt (merah)



Gambar 3.2 Contoh Jari yang Berputar pada Dataset 2_1.tif.txt (biru) & 2_2.tif.txt (merah)

Dari hasil observasi tersebut dapat disimpulkan bahwa adanya sifat kelabilan posisi titik *minutiae* yang sering berubah karena posisi jari saat melakukan proses *scanning*. Hal ini menyebabkan data koordinat yang asli tidak bisa sebagai data yang langsung bisa diverifikasi antara *template* dan *query*. Perlu adanya pembuatan *template* untuk tiap titik *minutiae* sebagai titik pusat sedangkan titik lainnya sebagai titik tetangga. Selanjutnya dihasilkan banyak *template* untuk setiap titik pusatnya tidak hanya satu *template* saja.

Terdapat beberapa data yang bisa diketahui yaitu titik *minutiae* (*x,y*) dan orientasi (φ) sebagai fitur yang bisa disimpan. Untuk tipe titik *minutiae* (*t*) tidak digunakan karena variasinya sedikit. Agar aman maka fitur yang disimpan tidak boleh fitur yang sudah diketahui dari dataset. Perlu adanya proyeksi garis dengan memanfaatkan titik koordinat dan orientasi dalam representasi data. Sehingga ketika melakukan verifikasi antara *template* dan *query* menggunakan representasi data yang memang unik agar *irreversible* dan lebih aman.

3.2 Data

Pada subbab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan.

3.2.1 Data Masukan

Data masukan adalah data awal yang akan dimasukkan ke dalam sistem. Data yang digunakan dalam perangkat lunak untuk proteksi sidik jari adalah *dataset* FVC2002DB2a dari *International competition for Fingerprint Verification Algorithms*[18]. Data berisi 100 pasang sidik jari, untuk masing-masing data merupakan data kumpulan titik *minutiae* dengan rincian koordinat posisi *minutiae* (*x,y*), orientasi dari *minutiae* (φ), serta tipe *minutiae* (*t*).

100 pasang data akan dibandingkan tingkat kesamaannya satu dengan lainnya sehingga terdapat 10.000 verifikasi.

Satu pasang data akan dijadikan sebagai *template* dan *query*. *Template* didapat dari awal *dataset* yang ditransformasi untuk disimpan di *database* sebagai data master. Sedangkan *Query* adalah hasil dari transformasi sidik jari yang akan dibandingkan dengan data master. Pada **Gambar 3.3** terdapat data yang terdiri dari 4 kolom yaitu titik koordinat *x* & *y*, orientasi *φ* dan tipe *t*. Masukkan data pada tugas akhir ini hanya menggunakan titik koordinat *x,y* dan orientasi *minutiae*. Tipe *minutiae* tidak digunakan karena kurang bervariasi hanya bernilai nol atau satu saja.

```
115   71   0.466330159517235   0
108   103   3.48520435007618   0
74   104   0.294524311274043   0
183   104   3.77972866135022   1
246   116   3.85335973916873   0
```

Gambar 3.3 Contoh data masukan FVC2002DB2a 1_1.txt

3.2.2 Data Keluaran

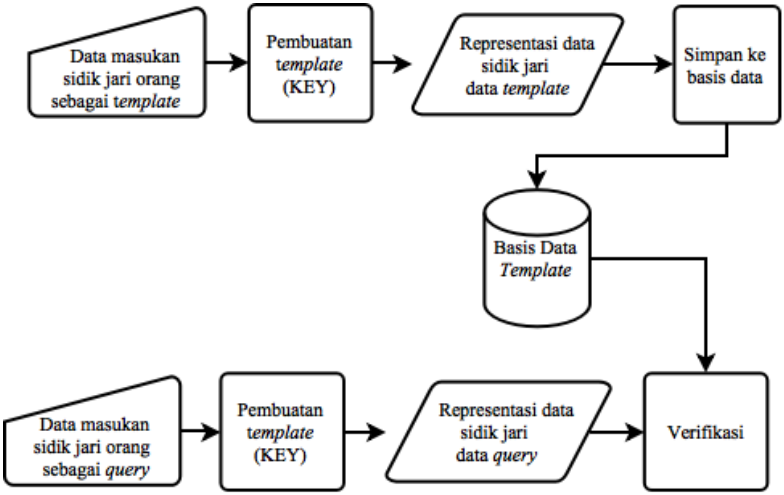
Data masukan akan diproses dengan menggunakan metode *cancelable template* dengan menggunakan *convex hull*. Hasil dari proses proteksi sidik jari adalah cocok atau tidak cocoknya data dengan *threshold* tertentu.

3.3 Desain Umum Sistem

Rancangan perangkat lunak proteksi sidik jari sesuai dengan metode *cancelable template* yang menitikberatkan kepada pembuatan *template/query* dan verifikasi. Secara umum mekanisme sistem *cancelable template* yang diusulkan dapat dilihat pada **Gambar 3.4** yang dijelaskan sebagai berikut:

1. Masukan berupa data sidik jari titik *minutiae* dari seseorang sebagai data *template*.
2. Pembuatan *template* pada data *template* dengan *key* tertentu.
3. Menghasilkan representasi data sidik jari yang *irreversible* kemudian disimpan di basis data.
4. Ketika seseorang ingin melakukan test sidik jari maka terdapat masukan berupa data sidik jari titik *minutiae* dari seseorang sebagai data *query*.
5. Dilakukan pembuatan *template* pada data *query* dengan *key* yang sama pada waktu membuat data *template*.
6. Menghasilkan representasi data sidik jari yang *irreversible* juga.
7. Lalu antara data *template* dan data *query* dilakukan verifikasi apakah cocok atau tidak.

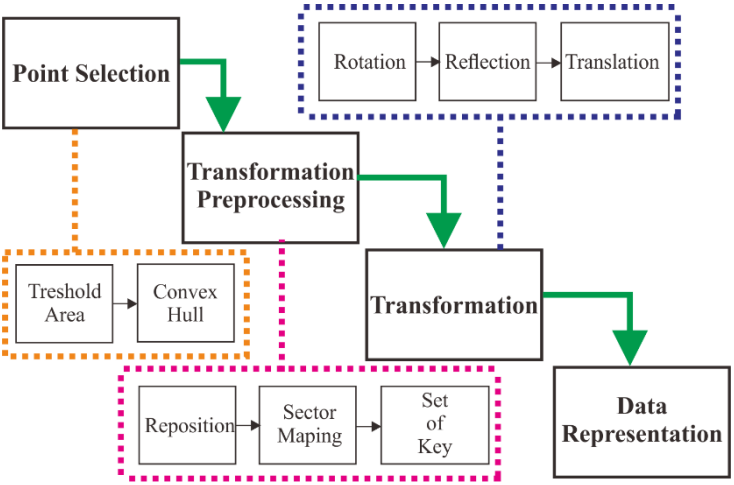
Mekanisme sistem *cancelable template* pada tugas akhir ini sesuai dengan mekanisme pada umumnya tentang *cancelable template*[6], [7]. Namun, dalam pengerjaannya terdapat modifikasi dan variasi yang ditunjukkan pada **Tabel 9.1**.



Gambar 3.4 Mekanisme Sistem *Cancelable Template* yang Diusulkan secara Umum

3.4 Perancangan Pembuatan *Template/Query*

Pada proses pembuatan *template* dilakukan metode secara umum yang ditunjukkan pada **Gambar 3.5** yang terdiri dari beberapa langkah yaitu seleksi titik (*point selection*), transformasi pra proses (*transformation preprocessing),* transformasi *(transformation),* dan representasi data (*data representation)*. Pada seleksi titik terdapat proses *threshold area* dan *convex hull.* Pada transformasi pra proses terdapat reposisi titik *minutiae*, pembagian sektor, dan perancangan *key*. Pada transformasi terdapat rotasi, refleksi, dan translasi. Hasil dari pembuatan *template* ini berupa *template* yang nantinya bisa dilakukan verifikasi antara data *template* dan *query*.



Gambar 3.5 Alur Metode Secara Umum

Berikut akan dijelaskan perancangan pembuatan *template* dari metode secara umum yang diusulkan pada tugas akhir ini secara rinci.

3.4.1 Perancangan Seleksi Titik

Tahap pertama diawali dengan proses menyeleksi titik yang akan digunakan untuk proses selanjutnya. Seleksi titik digunakan untuk menyeleksi beberapa titik yang nantinya akan dibuat titik pusat membuat *template*. Dalam proses seleksi titik terbagi menjadi dua langkah yaitu diawali dengan menyeleksi sesuai *threshold area* dan dilanjutkan dengan metode *convex hull*.

3.4.1.1 *Threshold Area*

Pada seleksi titik awal sesuai dengan *threshold area* dilakukan dengan cara mencari selisih jarak terjauh dan selisih orientasi terjauh dengan *threshold* tertentu. Pencarian ini dilakukan dengan membandingkan satu titik dengan titik lainnya. Pada selisih jarak terjauh dihitung menggunakan *ecluidean distance* dengan **Persamaan (3.1)**, dimana $\Delta\, r$ adalah selisih jarak terjauh, x dan y adalah koordinat x dan y titik *minutiae*. Pada selisih orientasi titik terjauh dapat dihitung dengan **Persamaan (3.2)**, $\Delta\, a$ adalah selisih orientasi terjauh, θ adalah orientasi titik *minutiae* φ dalam radian. Selanjutnya pada **Persamaan (3.3)** akan dihasilkan *distance* dua titik m_1 dan m_2 dengan *threshold* $t_1 = 1$ dan $t_2 = 0.2$ [6]. Dengan didapatkan dua titik terjauh yaitu m_1 dan m_2 maka dapat dicari koordinat titik pusat bayangan untuk titik x (x_{core}) pada **Persamaan (3.4)** dan titik y (y_{core}) pada **Persamaan (3.5)** serta radius bayangan (r_{core}) pada **Persamaan (3.6)**.

$$\Delta\, r = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{3.1}$$

$$\Delta\, a = \min(\left|\theta_i - \theta_j\right|, (360 - \left|\theta_i - \theta_j\right|)) \tag{3.2}$$

$$dis\left(m_i, m_j\right) = t_1\, x\, \Delta\, r + t_2\, x\, \Delta\, a \tag{3.3}$$

$$x_{core} = \frac{x_{m1} + x_{m2}}{2} \tag{3.4}$$

$$y_{core} = \frac{y_{m1} + y_{m2}}{2} \tag{3.5}$$

$$r_{core} = \frac{\sqrt{(x_{m1} - x_{m2})^2 + (y_{m1} - y_{m2})^2}}{2} \tag{3.6}$$

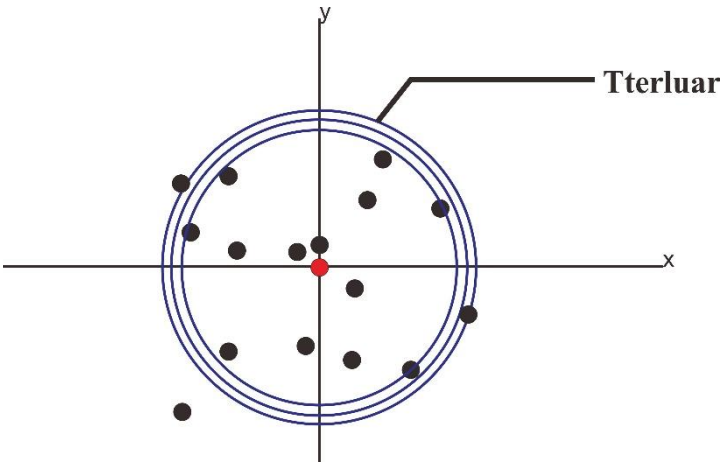
3.4.1.2 Convex Hull

Setelah pencarian area *threshold* untuk mendapatkan titik pusat bayangan (m_{core}) dan radius bayangan (r_{core}), maka dilanjutkan dengan seleksi titik menggunakan *convex hull* . Seperti yang sudah dilakukan *convex hull* pada referensi paper[7] menggunakan *graham scan.* Pada penelitian ini menggunakan *Jarvis March*[12]. Dikarenakan algoritma *jarvis march* lebih cepat dan sama akurat dibandingkan dengan algoritma *graham scan* pada *convex hull* [20].

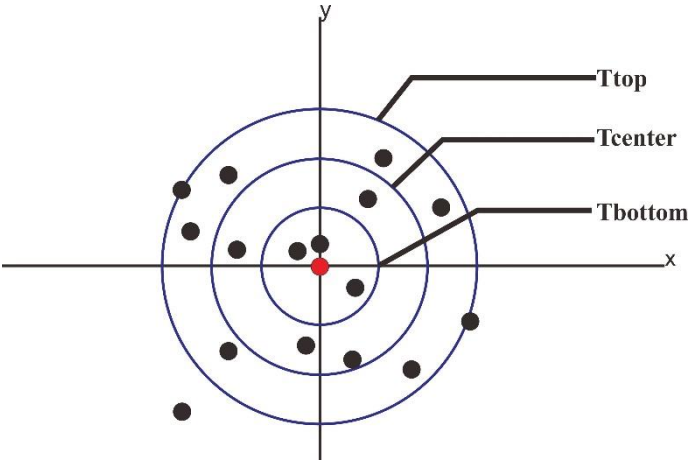
Pada referensi paper [7] dijelaskan untuk seleksi titik dilakukan *convex hull* sebanyak tiga kali untuk area terluar yang dapat dilihat pada **Gambar 3.6.** Namun, pengambilan area terluar sebanyak tiga kali kurang tepat karena area terluar dari *dataset* sering berubah-ubah yang disebabkan pergeseran peletakan sidik jari saat proses *scanning* oleh pengguna. Oleh karena itu, perlu adanya pengambilan titik pada setiap area agar mengurangi pencocokan palsu. Pada tugas akhir ini penulis menggunakan tiga area yang dapat dilihat pada **Gambar 3.7** yang terdiri dari T_{top} (area luar) menggunakan $3/3\ r_{core}$, T_{center} (area tengah) menggunakan $2/3\ r_{core}$, dan T_{bottom} (area dalam) menggunakan $1/3\ r_{core}$. Langkah-langkah seleksi titik yang diterapkan adalah sebagai berikut:

1) Lakukan *convex hull* pada area luar. Simpan titik hasil *jarvis march* pada penyimpanan data *convex hull.*

2) Lakukan *convex hull* pada area tengah. Simpan titik hasil *jarvis march* pada penyimpanan data *convex hull.*
3) Lakukan *convex hull* pada area dalam. Simpan titik hasil *jarvis march* pada penyimpanan data *convex hull.*



Gambar 3.6 *Convex Hull* Terluar



Gambar 3.7 *Convex Hull* Tiga Area

3.4.2 Perancangan Transformasi Pra Proses

Tahap kedua dilanjutkan dengan transformasi pra proses yaitu persiapan data titik *minutiae* sebelum dilakukan transformasi dan direpresentasikan. Dalam perancangan transformasi pra proses ini terbagi menjadi tiga tahap yaitu reposisi titik *minutiae*, pemetaan sektor, dan perancangan *key*.

3.4.2.1 Reposisi Titik *Minutiae*

Pada waktu transformasi dan representasi data dibutuhkan titik pusat sebagai pusat transformasi maupun titik pusat untuk melakukan proyeksi garis. Titik pusat bayangan tidak bisa dijadikan titik pusat karena titik *minutiae* terkadang terjadi perpindahan posisi sidik jari oleh pengguna pada saat proses *scanning*. Titik pusat diperoleh dari titik *minutiae* yang sudah terseleksi sesuai observasi sebelumnya pada **Subbab 3.1**. Sehingga akan dilakukan perulangan pembuatan *template* sesuai jumlah titik *minutiae* yang terseleksi pada tahap seleksi titik.

Pada setiap titik pusat yang terpilih akan dilakukan penyesuaian titik *minutiae* lain (titik tetangga) terhadap titik pusat yang dalam hal ini disebut proses reposisi titik *minutiae*. Reposisi titik *minutiae* akan dilakukan untuk semua titik tetangga terhadap titik pusat yang akan menghasilkan koordinat baru dan sudut orientasi baru. Langkah-langkah reposisi titik *minutiae* adalah sebagai berikut:

1) Koordinat titik pusat terpilih P_i yang berisi (x_{pusat}, y_{pusat}) serta orientasi θ_{pusat} akan menjadi titik pusat koordinat pada titik (0,0).
2) Garis yang dibentuk oleh titik pusat terpilih dengan sumbu $y_{pusat} = 0$ akan dijadikan sumbu x yang baru, sedangkan untuk sumbu y, dibentuk dengan membuat garis tegak lurus terhadap sumbu x yang telah dibuat sebelumnya.

3) Merotasikan titik tetangga terhadap titik pusat sebesar orientasi titik pusat yang dipilih θ_{pusat} dengan arah berlawanan arah jarum jam.
4) Hasil koordinat baru titik tetangga dihitung dengan **Persamaan (3.7)**, dimana (x_i, y_i) adalah koordinat titik tetangga dan (x'_i, y'_i) adalah koordinat titik baru hasil transformasi pra proses (P'_i) . Sedangkan orientasi baru dihitung dengan **Persamaan (3.8)**, dimana θ adalah orientasi titik tetangga dan θ' adalah orientasi baru hasil transformasi pra proses.

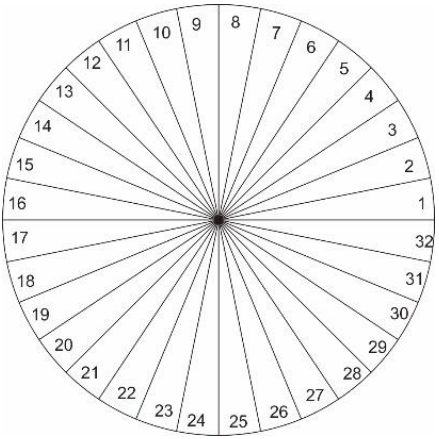
$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} \cos\theta_{pusat} & -\sin\theta_{pusat} \\ \sin\theta_{pusat} & \cos\theta_{pusat} \end{bmatrix} * \begin{bmatrix} x_i - x_{pusat} \\ y_i - y_{pusat} \end{bmatrix} \quad (3.7)$$

$$\theta' = mod\ (\theta - \theta_{pusat}, 360) \quad (3.8)$$

3.4.2.2 Pemetaan Sektor

Pada proses transformasi pra proses juga dilakukan *sector mapping*. Tujuannya untuk mengelompokkan titik *minutiae* menjadi beberapa kelompok sektor sesuai dengan koordinat titik tetangga dari posisi titik pusatnya. Pengelompokan titik *minutiae* dilakukan untuk proses rotasi dan refleksi pada tahap transformasi. Pada tugas akhir ini digunakan sektor berjumlah 32. Pengelompokan sektor didasarkan pada besar sudut *polar* yang dihasilkan dengan sumbu *x* positif. Besar sudut satu sektor adalah 11.25^o yang dihasilkan dari pembagian 360^o dengan jumlah sektor yaitu 32. Berikut gambaran sektor dapat dilihat pada **Gambar 3.8.**

Untuk memetakan suatu titik ke dalam sebuah sektor dihasilkan dari nilai *arctan* yang disesuaikan dengan letak titik yang ada. Pada kuadran 1 dan kuadran 3 besar sudut yang dibentuk oleh *arctan* yaitu sudut α, sedangkan pada kuadran 2 dan kuadran 4 besar sudutnya 90^o – α. Dimana α adalah hasil yang dihitung dari sudut yang terbentuk dengan sumbu *x*.



Gambar 3.8 Pembagian dan Letak Sektor

3.4.2.3 Perancangan *Key*

Dalam metode proteksi sidik jari menggunakan *cancelable template* terdapat fitur dasar yaitu adanya sebuah *key.* Tujuan *key* ini adalah untuk mendapatkan *template* yang berbeda-beda dari sebuah data sidik jari. Sehingga dihasilkan *template* yang aman. Misalkan, jika *key* yang membentuk data *template* dicuri, maka data *template* akan dibuat baru lagi dengan *key* yang baru juga. Cara ini lebih aman dibandingkan menyimpan data sidik jari langsung, karena sidik jari seseorang hanya satu data saja ketika dicuri maka tidak bisa dibuat lagi layaknya *password* yang bisa diubah-ubah. Pada metode yang digunakan, sebuah *key* akan direpresentasikan kedalam sebuah matriks.

Representasi *key* dapat dilihat pada **Persamaan (3.9)** dengan penjelasan sebagai berikut:

1) Pada sebuah *key* (*K*) terdapat 32 buah baris yang menunjukkan jumlah sektor dimana i adalah *key indeks* dan B_i adalah matriks *key.*
2) Pada setiap matriks B_i terdapat tiga nilai matriks yang berisi Ss_i, Sx_i, dan Sy_i.

3) Kolom pertama (Ss_i) menunjukkan pergeseran rotasi sektor dalam satuan sektor, misalkan 3 sektor, 6 sektor, dan sebagainya.
4) Kolom kedua (Sx_i) menunjukkan pergeseran terhadap sumbu *x* untuk proses translasi.
5) Kolom ketiga (Sy_i) menunjukkan pergeseran terhadap sumbu *y* untuk proses translasi.

$$\begin{cases} K = \{B_i\}_{i=1}^{32} \\ B_i = \{Ss_i, Sx_i, Sy_i\} \end{cases} \quad (3.9)$$

Key yang dibentuk kemungkinan ditemukan kombinasi yang melemahkan sisi keamanan yang biasanya dikenal dengan istilah *weak key.* Untuk mengatasinya maka dalam pembuatan *key* tidak boleh bernilai 0 dan tidak boleh kelipatan dari jumlah sektor yaitu 32 .

3.4.3 Perancangan Transformasi

Tahap ketiga yaitu transformasi yang akan dilakukan perpindahan titik-titik *minutiae* menggunakan transformasi geometri yang berupa rotasi, refleksi, dan juga translasi. Sebelumnya sudah dilakukan pengelompokan sektor dan pembuatan *key*, maka untuk selanjutnya setiap titik *minutiae* mempunyai *key index*, jumlah pergeseran *sektor,* dan juga pergeseran titik *x* dan *y* yang berbeda-beda sesuai kelompok sektornya. Dengan adanya transformasi akan menghasilkan koordinat titik dan orientasi baru yang bisa dilihat pada **Persamaan (3.10)**, untuk P_i adalah koordinat titik awal asli, P'_i adalah koordinat titik setelah transformasi pra proses, P''_i adalah koordinat titik setelah rotasi, P'''_i adalah koordinat titik setelah refleksi, dan P''''_i adalah koordinat titik setelah translasi. Untuk orientasi (θ) juga berubah kecuali pada waktu translasi tidak ada perubahan jadi perubahan orientasi yang paling terakhir adalah setelah refleksi.

$$\begin{cases} P_i = \{x_i, y_i\} \\ P'_i = \{x'_i, y'_i\} \\ P''_i = \{x''_i, y''_i\} \\ P'''_i = \{x'''_i, y'''_i\} \\ P''''_i = \{x''''_i, y''''_i\} \end{cases} \quad (3.10)$$

Berikut akan dijelaskan perancangan transformasi yang terdiri dari rotasi, refleksi dan translasi.

3.4.3.1 Rotasi

Pada transformasi rotasi, titik tetangga hasil dari transformasi pra proses akan dirotasi sesuai *key* rotasi sektor yang sudah dipetakan. Koordinat titik setelah rotasi P''_i didapatkan dengan **Persamaan (3.11)**, dimana (x'_i, y'_i) merupakan titik *minutiae* hasil dari transformasi pra proses dan titik *minutiae* akan dirotasi sejauh sudut ω yang diberikan. Untuk mendapatkan sudut ω yang digunakan untuk merotasi didapat dari **Persamaan (3.12)**, dimana Ss_i adalah pergeseran sektor. Misalkan sebuah titik *minutiae* berada pada sektor 2 dan *key sector* yang diberikan adalah 6, maka titik itu akan dirotasi dari sektor 2 ke sektor 8 dengan sudut 11.25° x 6 = 67,5°. Selain itu, orientasi sudut titik *minutiae* juga berubah sesuai dengan **Persamaan (3.13),** dimana θ' adalah orientasi setelah transformasi pra proses dan θ''adalah orientasi setelah rotasi hasil penambahan sebesar sudut ω.

$$\begin{bmatrix} x''_i \\ y''_i \end{bmatrix} = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} * \begin{bmatrix} x'_i \\ y'_i \end{bmatrix} \quad (3.11)$$

$$\omega = 11{,}25° \; x \; Ss_i \quad (3.12)$$

$$\theta'' = mod\,(\theta' + \omega, 360) \quad (3.13)$$

3.4.3.2 Refleksi

Berikutnya adalah tahap transformasi refleksi, refleksi ini dilakukan terhadap sumbu *x* atau refleksi terhadap sumbu *y*. Pemilihan refleksi terhadap sumbu *x* atau sumbu *y* berdasarkan ganjil genapnya sektor. Apabila sektor genap maka akan direfleksikan terhadap sumbu *x*. Sebaliknya apabila sektor ganjil maka akan direfleksikan terhadap sumbu *y*. Dengan **Persamaan (3.14)**, dimana $\mathrm{mod}(Ss_i, 2) = 0$ adalah ketika sektor genap maka direfleksikan terhadap sumbu *x*, dan $\mathrm{mod}(Ss_i, 2) = 1$ hasil ganjil maka direfleksikan terhadap sumbu *y*, sehingga menghasilkan koordinat *x*,*y* baru. Untuk orientasi juga berubah dengan **Persamaan (3.15)**, dimana θ''' adalah orientasi setelah refleksi dan θ'' orientasi setelah rotasi.

$$\begin{bmatrix} x'''_i \\ y'''_i \end{bmatrix} = \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} * \begin{bmatrix} x''_i \\ y''_i \end{bmatrix}, \mathrm{mod}(Ss_i, 2) = 0 \\ \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} x''_i \\ y''_i \end{bmatrix}, \mathrm{mod}(Ss_i, 2) = 1 \end{cases} \tag{3.14}$$

$$[\theta'''] = \begin{cases} 360 - \theta'', \mathrm{mod}(Ss_i, 2) = 0, & 0 \leq \theta'' \leq 360 \\ 180 - \theta'', \mathrm{mod}(Ss_i, 2) = 1, & 0 \leq \theta'' \leq 180 \\ 540 - \theta'', \mathrm{mod}(Ss_i, 2) = 1, & 180 < \theta'' \leq 360 \end{cases} \tag{3.15}$$

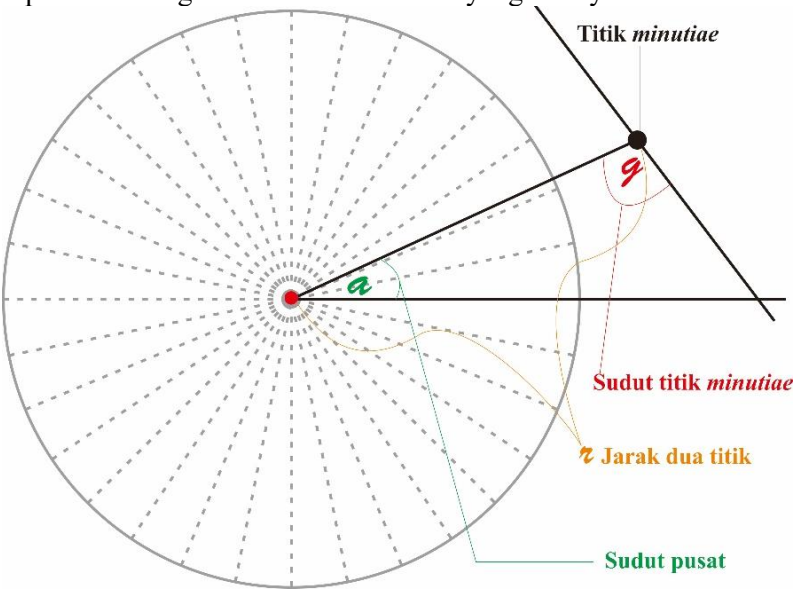
3.4.3.3 Translasi

Terakhir tahap translasi, seluruh titik *minutiae* akan digeser sesuai dengan **Persamaan (3.16)**, dimana Sx_i adalah *key* pergeseran koordinat *x,* Sy_i adalah pergeseran koordinat *y*, dan menghasilkan koordinat *x,y* baru. Sedangkan untuk orientasi tetap menggunakan orientasi setelah refleksi. Pada setiap sektor, translasi akan berbeda-beda sesuai dengan *key* yang diberikan. Tidak ada batasan *template* akan melebar jauh setelah proses transformasi, karena ketika dibatasi dengan *maksimum distance limitation* [7] banyak titik yang harusnya beda jadi sama dan terjadi pencocokan palsu.

$$\begin{bmatrix} x''''_i \\ y''''_i \end{bmatrix} = \begin{bmatrix} Sx_i \\ Sy_i \end{bmatrix} + \begin{bmatrix} x'''_i \\ y'''_i \end{bmatrix} \tag{3.16}$$

3.4.4 Perancangan Representasi Data

Tahap keempat yaitu representasi data yang akan dilakukan penyimpanan *template* yang bersifat *irreversible* ke dalam basis data. Fungsi *irreversible* adalah ketika suatu data yang sudah diproses tidak dapat kembali ke data asal. Implementasi kecil dari *irreversible* lainnya adalah ketika waktu seleksi titik, dimana titik yang digunakan tidak semua yang digunakan untuk membuat *template* tetapi sebagaian saja. Sehingga selain *variable* yang digunakan untuk representasi data seleksi titik juga membuat *template* menjadi *irreversible*. Representasi data ini lebih kepada enkapsulasi informasi yaitu menyembunyikan data yang diperlukan dengan membuat ke bentuk yang lainnya.



Gambar 3.9 Data Representasi

Pada tugas akhir ini terdapat tiga nilai *variable* yang dapat dilihat pada **Gambar 3.9** sebagai representasi data. Tiga nilai *variable* tersebut yaitu sudut pusat, jarak dua titik, dan sudut titik *minutiae*. Tiga nilai *variable* tersebut didapatkan dari proyeksi garis yang dibuat dari data-data titik *minutiae* hasil dari tahap transformasi seperti titik koordinat dan arah orientasi. Untuk tipe titik *minutiae* sendiri tidak digunakan karena nilainya boolean hanya 1 atau 0 sehingga kurang bagus digunakan sebagai perbandingan antar data. Tiga nilai *variable* sebagai representasi data akan dijelaskan sebagai berikut:

1) Sudut pusat (α)
   Sudut pusat didapat dengan membuat garis proyeksi dari titik pusat ke arah sumbu *x* positif dimana *y* = 0 dan juga garis proyeksi dari titik pusat ke arah titik *minutiae*. Kedua garis proyeksi tersebut, akan bertemu pada titik pusat yang selanjutnya dapat dicari sudut pusatnya. Sudut pusat (α) dapat dihitung dengan **Persamaan (3.17)**. Pada **Persamaan (3.17)**, α mewakili sudut pusat, y'''' koordinat *y* titik *minutiae* setelah proses translasi, dan x'''' koordinat *x* titik *minutiae* setelah proses translasi.

$$\alpha = \arctan\left(\frac{y''''}{x''''}\right) \tag{3.17}$$

2) Jarak dua titik (*r*)
   Jarak dua titik didapat dengan membuat garis proyeksi dari titik pusat ke arah titik *minutiae*. Garis proyeksi yang menghubungkan dua titik (*r*) dapat dihitung dengan **Persamaan (3.18)**. Pada **Persamaan (3.18)**, *r* mewakili jarak antara titik *minutiae* dengan titik pusat setelah ditransformasi. y'''' koordinat *y* titik *minutiae* setelah proses transformasi, dan x'''' koordinat *x* titik *minutiae* setelah proses transformasi.

$$r = \sqrt{(x'''')^2 + (y'''')^2} \tag{3.18}$$

3) Sudut titik *minutiae* (γ)
Sudut titik *minutiae* didapat dengan membuat garis proyeksi dari titik *minutiae* ke arah titik pusat dan juga garis proyeksi dari titik *minutiae* sesuai sesuai arah orientasi titik *minutiae* sampe bertemu dengan garis y=0. Sudut titik *minutiae* (γ) dapat dihitung dengan **Persamaan (3.19)**, terdapat dua persamaan untuk kuadran I & III dan kuadran II & IV dimana θ''' mewakili orientasi setelah refleksi, α mewakili sudut pusat, dan γ mewakili sudut titik *minutiae*.

$$\gamma = \begin{cases} mod\ (\theta''' ,180) - mod(\alpha, 180), \text{kuadran I dan III} \\ mod(\alpha, 180) - mod\ (\theta''' ,180), \text{kuadran II dan IV} \end{cases} \quad (3.19)$$

Pada proses transformasi pra proses pemilihan titik pusat menggunakan titik yang didapat dari hasil seleksi titik. Sehingga akan terdapat banyak titik pusat sesuai dengan jumlah hasil seleksi titik dan menghasilkan banyak data *template*. Hal ini, dikarenakan untuk mengantisipasi dari sifat kelabilan titik *minutiae* yang kadang muncul atau hilang. Akan berbahaya apabila menggunakan satu *template* atau satu titik pusat ketika titik tersebut ada pada data *template* tetapi hilang pada data *query* atau sebaliknya. Sedangkan untuk jumlah data masing-masing *template* sebanyak semua titik dikurangi satu. Misalkan total ada 10 titik, dan yang terseleksi ada 5 titik maka akan tercipta 5 *template,* dengan masing-masing *template* menampung 9 data, demikian juga *query*.

Pada **Gambar 3.10** mendeskripsikan jumlah *template* yang dihasilkan sebelum dan sesudah transformasi dan dapat dilihat juga representasi data yang dilakukan kedalam *template* baru. Terdapat *template* asli berjumlah 4 titik. Untuk masing masing-masing titik terdiri dari data (*x,y, φ,t*) yang terseleksi maupun tidak terseleksi pada waktu tahap seleksi titik. Dua data titik terseleksi menghasilkan dua *template* yaitu *Template* 1 dan *Template* 2. Untuk masing-masing *template* yang dihasilkan terdapat 3 baris data yang direpresentasikan dalam bentuk tuple yaitu (*α,r, γ*).

Template Asli				Seleksi Titik
x1	y1	φ1	t1	Titik terseleksi
x2	y2	φ2	t2	Titik tereseleksi
x3	y3	φ3	t3	Titik tidak terseleksi
x4	y4	φ4	t4	Titik tidak terseleksi

Template 1			*Template* 2		
α1	r1	γ1	α1	r1	γ1
α2	r2	γ2	α2	r2	γ2
α3	r3	γ3	α3	r3	γ3

Gambar 3.10 Template Hasil Transformasi

3.5 Perancangan Verifikasi Lokal dan Global

Selanjutnya *template* data akan dibandingkan dengan *query,* proses ini disebut proses verifikasi. Proses verifikasi terbagi menjadi dua yaitu verifikasi lokal dan verifikasi global [9]. Pada proses verifikasi secara lokal, akan dilakukan pengecekan pada isi sebuah *template* dan *query*. Sedangkan pada verifikasi secara global adalah membandingkan *template* dengan *query* secara umumnya. Mekanisme verifikasi dapat dilihat pada **Gambar 3.11** dengan penjelasan sebagai berikut:

1) Mencari semua kombinasi yang ada dengan memasangkan seluruh titik yang ada antara *query* dan *template*.
2) Setiap kombinasi dicari selisih α,r, dan γ dengan **Persamaan (3.20)**, **Persamaan (3.21),** dan **Persamaan (3.22)**. Pada **Persamaan (3.20)**, αT adalah nilai sudut pusat dari *template*, αQ adalah nilai sudut pusat dari *query*, dan s_α adalah selisih sudut pusat. Pada **Persamaan (3.21)**, rT adalah jarak dua titik dari *template*, rQ adalah jarak dua titik dari *query*, dan s_r adalah selisih jarak dua titik. Pada **Persamaan (3.22)**, γT

adalah nilai sudut titik *minutiae* dari *template*, γQ adalah nilai sudut titik *minutiae* dari *query*, dan s_γ adalah selisih sudut titik *minutiae*.

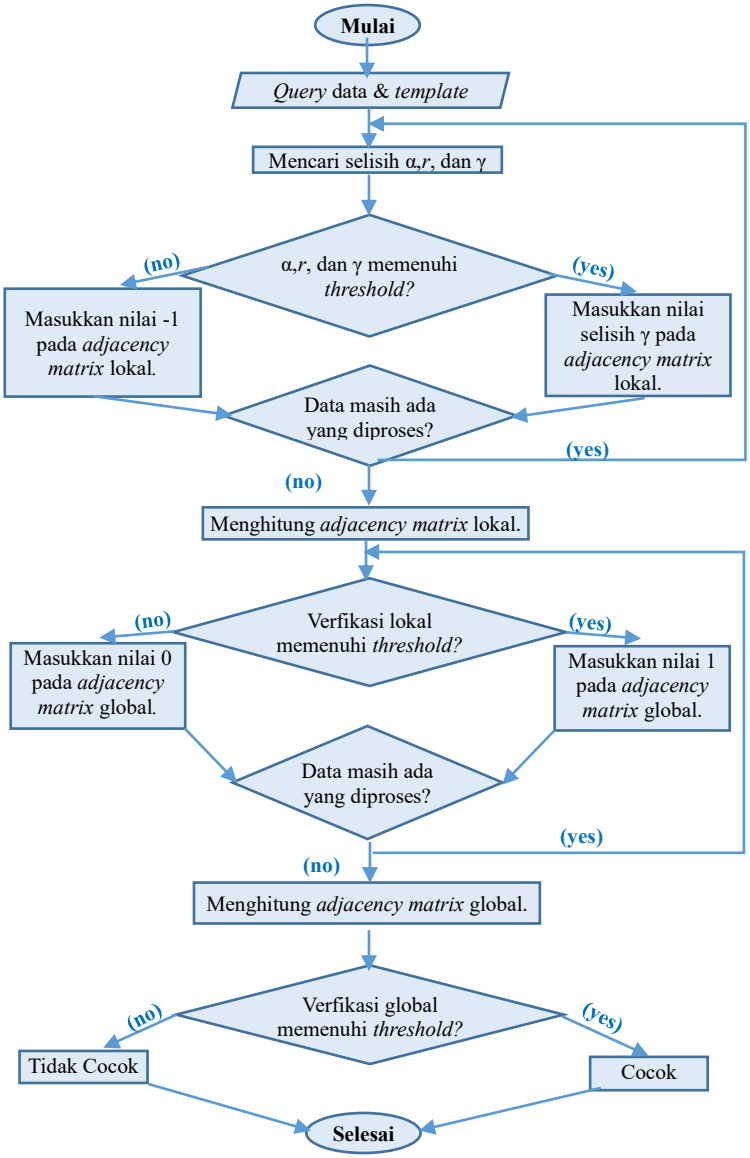
3) Lalu dilakukan pengecekan dengan **Persamaan (3.23)** apakah selisih dari sudut pusat, jarak dua titik, dan sudut titik *minutiae* memenuhi *threshold* yang diberikan. Pada **Persamaan (3.23)**, dimana t_1 adalah *threshold* sudut pusat, t_2 adalah *threshold* jarak dua titik, t_3 adalah *threshold* sudut titik *minutiae,* dan f adalah hasil pengecekan.
4) Jika pada waktu pengecekan memenuhi *threshold* maka bisa dituliskan selisih sudut titik *minutiae* (s_γ) pada *adjacency matrix* lokal dengan baris dan kolom sesuai nomor urut. Apabila tidak memenuhi *threshold* maka *adjacency matrix* lokal dapat diisi dengan nilai -1.
5) Setelah itu dilakukan penghitungan terhadap *adjacency matrix* lokal apakah akan melebihi *threshold* verifikasi lokal.
6) Jika memenuhi *threshold* maka *adjacency matrix* global diisi nilai 1 jika tidak dapat diisi nilai 0.
7) Pada verifikasi global akan dilakukan penghitungan *adjacency matrix* global.
8) Apabila pada verifikasi global penghitungan dapat memenuhi *threshold* verifikasi global maka dapat dikatakan *template* dan *query* tersebut cocok.

$$s_\alpha = \alpha T - \alpha Q \tag{3.20}$$

$$s_r = rT - rQ \tag{3.21}$$

$$s_\gamma = \gamma T - \gamma Q \tag{3.22}$$

$$f = \begin{cases} -1, & s_\alpha > t_1, s_\mathrm{r} > t_2, s_\gamma > t_3 \\ s_\gamma, & s_\alpha \leq t_1, s_\mathrm{r} \leq t_2, s_\gamma \leq t_3 \end{cases} \tag{3.23}$$



Gambar 3.11 Flowchart Mekanisme Verifikasi

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa *pseudocode* untuk membangun program.

4.1 Lingkungan Implementasi

Implementasi pengembangan proteksi sidik jari dengan metode *cancelable template* menggunakan *convex hull* dan garis proyeksi menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada **Tabel 4.1**.

Tabel 4.1 Spesifikasi Lingkungan Implementasi

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i3-3240 CPU @ 3.40GHx (4CPUs), ~3.4 GHz
	Memori	4 GB 798.1 MHz DDR3
	VGA	AMD Radeon HD 8470 Graphics, Memory 2840 MB
Perangkat Lunak	Sistem Operasi	Windows 10 Pro 64 bit
	Perangkat Pengembang	MATLAB R2013a
	Bahasa Pemrograman	Matlab

Selain itu, pada tugas akhir ini dalam pembuatan Grafik dan beberapa pengolahan angka didukung dengan *software Microsoft Excel*.

4.2 Implementasi Tahap Pembuatan *Template/Query*

Implementasi tahap pembuatan *template/query* dilakukan untuk membuat *template* yang disimpan pada basis data dan bersifat *irreversible*. Tahap pembuatan *template/query* pada tugas akhir ini terdiri dari dua tahapan besar. Pertama adalah seleksi titik (*threshold area, convex hull*). Kedua adalah transformasi yang terdiri dari transformasi pra proses, transformasi (rotasi, refleksi, translasi), dan representasi data menggunakan proyeksi garis. *Pseudocode* pembuatan *template* terdapat pada fungsi *makeTemplate* yang dapat dilihat pada **Gambar 4.2**. Masukan fungsi *makeTemplate* berupa nama *file* dari data masukan titik *minutiae*. Pada baris ke 5 dilakukan pembacaan file dengan fungsi *readFile* sesuai nama *file* lalu mengembalikan *total* jumlah titik *minutiae* dan *matrix* merupakan *array* dari data masukan titik *minutiae*.

```
1  ALGORITMA MakeTemplate( namafile )
2  //MakeTemplate : membuat template dari query/template
3  // Input : nama file template/query
4  // Output : file template/query akan disimpan
5  [total, matrix] ← ReadFile(namafile)
6  selectionPoint ← []
7  selectionPoint ← SelectionPoint(total,matrix)
8  Transformasi(selectionPoint,matrix,strcat('DATA/COBA/',
   namafile))
```

Gambar 4.1 *Pseudocode* Fungsi *MakeTemplate*

Selanjutnya dilakukan seleksi titik dan transformasi dengan menggunakan fungsi *SelectionPoint* pada baris 7 dan *Transformasi* pada baris 8. Penjelasan dari masing-masing tahapan implementasi pembuatan *template* yaitu seleksi titik dan transformasi dijelaskan pada **Subbab 4.2.1**, dan **Subbab 4.2.2**.

4.2.1 Implementasi Seleksi Titik

Seleksi titik digunakan untuk mengurangi titik yang digunakan sebagai titik pusat dalam pembuatan *template/query. Pseudocode* seleksi titik terdapat pada fungsi *SelectionPoint* yang dapat dilihat pada **Gambar 4.2**. Data masukan berupa *total* yang berupa jumlah titik *minutiae* dan *matrix* merupakan *array* semua titik *minutiae*. Sedangkan data keluaran fungsi *SelectionPoint* adalah *selectionPoint* yang merupakan titik hasil seleksi titik. Pada fungsi *SelectionPoint* memanggil fungsi *Threshold Area* dan juga *ConvexHullArea.* Dua fungsi tersebut merupakan dua tahapan pada seleksi titik yaitu *threshold area* dan *convex hull* yang akan dijelaskan lebih lanjut pada **Subbab 4.2.1.1**, dan **Subbab 4.2.1.2**.

```
1  ALGORITMA SelectionPoint(total,matrix)
2  //SelectionPoint : Terdiri dari seleksi threshold area
   & convex hull
3  // Input : matrix adalah array semua titik, total
   adalah jumlah array
4  // Output : array hasil seleksi point
5  selectionPoint ← []
6  [r_core,x_core,y_core] ← ThresholdArea(total,matrix)
7  selectionPoint ← ConvexHullArea
   (total,matrix,x_core,y_core,r_core)
8  return selectionPoint
```

Gambar 4.2 *Pseudocode* Fungsi *SelectionPoint*

4.2.1.1 Implementasi *Threshold Area*

Pada implementasi *threshold area* dilakukan seleksi titik sesuai *threshold* yang diberikan dan pembuatan titik pusat bayangan untuk pembagian area. *Pseudocode threshold area* terdapat pada fungsi *ThresholdArea* yang dapat dilihat pada **Gambar 4.3**. Data masukan berupa *total* yang berupa jumlah titik *minutiae* dan *matrix* merupakan *array* semua titik *minutiae*. Sedangkan data keluaran fungsi *ThresholdArea* adalah koordinat titik pusat bayangan (*x_core, y_core*) dan jari-jarinya *r_core*.

```
1  ALGORITMA ThresholdArea( total,matrix)
2  //ThresholdArea : Untuk mencari thresholdArea dan
   mencari titik pusat
3  //bayangan serta jari-jari yang dibentuk
4  // Input : matrix adalah array semua titik, total
   adalah jumlah array
5  // Output : titik x_core,y_core dan jari-jari r_core
6  matrix_dist ← []
7  for i ← 1 to total do
8      for j ← i+1 to total do
9          deltaR ← sqrt((matrix(j,2)-matrix(i,2))^2 +
   (matrix(j,3)-matrix(i,3))^2)
10         sudut1 ← matrix(j,4)/pi*180
11         sudut2 ← matrix(i,4)/pi*180
12         deltaA ← min([abs(sudut1-sudut2)360-
   (abs(sudut1-sudut2))])
13         distance ← deltaR*1 + deltaA*0.2
14         matrix_dist ← [matrix_dist;[i,j,distance]]
15 [M,I] ← max(matrix_dist(:))
16 [I_row, I_col] ← ind2sub(size(matrix_dist),I)
17 titik1 ← matrix_dist(I_row,1)
18 titik2 ← matrix_dist(I_row,2)
19 r_core ← ceil(sqrt((matrix(titik1,2)-
   matrix(titik2,2))^2 + (matrix(titik1,3)-
   matrix(titik2,3))^2)/2)
20 x_core ← 1/2*(matrix(titik1,2)+matrix(titik2,2))
21 y_core ← 1/2*(matrix(titik1,3)+matrix(titik2,3))
22 return [x_core,y_core,r_core]
```

Gambar 4.3 *Pseudocode* Fungsi *ThresholdArea*

Pada fungsi *ThresholdArea* baris 7 hingga 14 untuk semua kombinasi titik dilakukan penghitungan jarak dua titik(*distance)* untuk dimasukkan pada *array matrix_dist*. Perhitungan *distance* pada baris 13 didapatkan dari menambahkan jarak titik terjauh(*deltaR)* dan orientasi terjauh(*deltaA)* yang dikalikan *threshold* tertentu[6]. Untuk *threshold deltaA* sebesar 0,2 sedangkan *threshold deltaR* sebesar 1,0. Jarak titik terjauh(*deltaR)* didapat dengan melakukan perhitungan *ecluidean distance* dari dua titik yang terdapat pada baris 9. Sedangkan *deltaA* didapat dari selisih *sudut1* dan *sudut2* yang terdapat pada baris 10 hingga 12.

Untuk masing-masing *sudut1* dan *sudut2* dilakukan perhitungan sudut dimana orientasi masih berbentuk nilai radian.

Selanjutnya pada fungsi *ThresholdArea* dilakukan pencarian maksimal *distance* pada baris 15 dan mendapatkan dua titik yaitu *titik1* dan *titik2* pada bari 17 hingga 18. Dari dua titik tersebut dapat diperoleh jari-jari pusat bayangan(*r_core*) pada baris 19 dan koordinat titik pusat bayangan (*x_core,y_core*) pada baris 20 hingga 21. Dan pada baris 22 atau baris terakhir dikembalikan nilai *x_core, y_core*, dan *r_core* pada fungsi *ThresholdArea*.

4.2.1.2 Implementasi *Convex Hull*

Pada implementasi seleksi titik menggunakan *convex hull* akan dilakukan *convex hull* pada tiga area yang sudah ditentukan. *Pseudocode convex hull* pada tiga area terdapat pada fungsi *ConvexHullArea* yang dapat dilihat pada **Gambar 4.4**. Data masukan berupa *total* yang berupa jumlah titik *minutiae*, *matrix* merupakan *array* semua titik *minutiae, (x_core, y_core*) merupakan koordinat titik pusat bayangan dan *r_core* sebagai jari-jari. Sedangkan data keluaran fungsi *ThresholdArea* adalah *selectionPoint* adalah titik hasil seleksi *convex hull* pada tiga area.

Pada fungsi *ThresholdArea* dilakukan inisiasi *selectionPoint* pada baris 6. Lalu untuk masing-masing area pada baris 7 hingga 9 dilakukan fungsi *ConvexHull*. Parameter pemanggilan fungsi *ConvexHull* hampir sama, yang berbeda hanya pada parameter terakhir untuk jari-jarinya. Pada *resultBottom* yang merupakan *convex hull* area dalam *(*T_{bottom}) menggunakan 1/3 jari-jari. Untuk *resultCenter* yang merupakan *convexhull* area tengah *(*T_{center}) menggunakan 2/3 jari-jari. Sedangkan pada *resultTop* yang merupakan *convexhull* area luar *(*T_{top}) menggunakan 3/3 jari-jari. Selanjutnya pada baris 10 hingga 15 dilakukan penyimpanan untuk masing-masing hasil fungsi *ConvexHull*. Lalu pada baris 16 dan 17 dilakukan pengurutan *indeks* dan penghapusan jika ada *indeks* titik *minutiae* yang sama agar tidak *redundant*. Nilai kembalian fungsi *ThresholdArea* yaitu *selectionPoint* sebagai hasil seleksi titik.

```
1  ALGORITMA ConvexHullArea
   (total,matrix,x_core,y_core,r_core)
2  //ConvexHullArea : Mencari convex hull pada tiga area
   yaitu top,center,bottom
3  // Input : matrix adalah array semua titik, total
   adalah jumlah array,
4  // x_core adalah koordinat x titik tengah x, y_core
   adalah koordinat y titik
5  // Output : selectionPoint adalah titik yang
   terseleksi
6  selectionPoint  ← []
7  resultBottom  ← ConvexHull(
   (matrix(:,1:4)),(total),(x_core),(y_core),(r_core/3) )
8  resultCenter  ← ConvexHull(
   (matrix(:,1:4)),(total),(x_core),(y_core),(r_core/3*2)
   )
9  resultTop  ← ConvexHull(
   (matrix(:,1:4)),(total),(x_core),(y_core),(r_core) )
10 for i  ← 1 to size(resultBottom,1) do
11     selectionPoint  ← [selectionPoint ;
   matrix(resultBottom(i,1),1:4)]
12 for i  ← 1 to size(resultCenter,1) do
13     selectionPoint  ← [selectionPoint ;
   matrix(resultCenter(i,1),1:4)]
14 for i  ← 1 to size(resultTop,1) do
15     selectionPoint  ← [selectionPoint ;
   matrix(resultTop(i,1),1:4)]
16 selectionPoint  ← sortrows(selectionPoint,1)
17 selectionPoint  ← unique(selectionPoint, 'rows')
18 return selectionPoint
```

Gambar 4.4 *Pseudocode* Fungsi *ConvexHullArea*

Pada fungsi *ConvexHull* sendiri diimplementasikan oleh *pseudocode* yang dapat dilihat pada **Gambar 4.5**. Data masukan berupa *total* yang berupa jumlah titik *minutiae*, *matrix* merupakan *array* semua titik *minutiae, (x_core, y_core*) merupakan koordinat titik pusat bayangan dan *r_core* sebagai jari-jari. Sedangkan data keluaran fungsi *ConvexHull* adalah *selectionPoint* adalah titik hasil seleksi *convex hull*. Pada fungsi *ConvexHull* baris 8 hingga 11 dilakukan seleksi lingkaran dengan fungsi *selectionCircle* untuk titik *minutiae* yang berada pada lingkaran dengan jari-jari sesuai area yang dihitung. Selanjutnya pada baris 12 dilakukan pencarian titik *minutiae* dengan fungsi *ConvexHullAlgorithm*.

```
1  ALGORITMA ConvexHull(
   matrix,total,x_core,y_core,r_core )
2  //ConvexHull : dimulai menyelksi area lingkaran
   (top,center,bottom) lalu
3  // mencari titik terluar dengan Convex Hull Algorithm
4  // Input : matrix adalah array semua titik, total
   adalah jumlah array,
5  // x_core adalah koordinat x titik tengah x, y_core
   adalah koordinat y titik
6  // tengah, r_core adalah jari-jari
7  // Output : result adalah array titik yang terseleksi
8  circle ← SelectionCircle(
   (matrix(:,1:3)),(total),(x_core),(y_core),(r_core))
9  titik ← []
10 for i ← [1:size(circle,1)]
11     titik ← [titik ; matrix(circle(i,1),1:3)]
12 result ← ConvexHullAlgorithm(titik,size(titik,1))
13 return result
```

Gambar 4.5 *Pseudocode* Fungsi *ConvexHull*

Pada fungsi *SelectionCircle* sendiri diimplementasikan oleh *pseudocode* yang dapat dilihat pada **Gambar 4.6**. Data masukan berupa *n* yang berupa jumlah titik *minutiae*, *m* merupakan *array* semua titik *minutiae, (a,b*) merupakan koordinat titik pusat bayangan dan *r* sebagai jari-jari. Sedangkan data keluaran fungsi *SelectionCircle* adalah *matrix* adalah titik hasil seleksi lingkaran.

```
1  ALGORITMA SelectionCircle( m,n,a,b,r )
2   //SelectionCircle : menyeleksi titik dengan lingkaran
   titik pusat dan
3   //jari2 tertentu
4   // Input : m adalah array semua titik, n adalah
   jumlah matrix, a adalah
5   // titik x, b adalah titik y, r adalah jari-jari
6   // Output : Array titik baru yang terseleksi dalam
   lingkaran
7  matrix ← [];
8  for i ← [1:n]
9      if(r^2 >=(m(i,2)-a)^2 + (m(i,3)-b)^2)
10         matrix ← [matrix;m(i,1)]
11 return matrix
```

Gambar 4.6 *Pseudocode* Fungsi *SelectionCircle*

Pada fungsi *ConvexHullAlgorithm* sendiri diimplementasikan oleh *pseudocode* yang dapat dilihat pada **Gambar 4.7**. Data masukan berupa *points* semua titik *minutiae* hasil seleksi lingkaran, *n* merupakan jumlah titik *minutiae*. Sedangkan data keluaran fungsi *ConvexHullAlgorithm* adalah *hull* yang merupakan titik hasil *convex hull*. Minimal ada 3 titik untuk melakukan *convex hull* yang terdapat pada baris 6. Selanjutnya baris 7 hingga 10 dilakukan pencarian titik koordinat yang terdapat pada paling kiri. Selanjunya pada bari 12 hingga 21 dilakukan pencarian titik terluar dengan memanggil fungsi *Orientation* untuk menghitung posisi sudut polar. Jika satu titik terluar terpilih maka titik disimpan pada *variable hull* dan diulang kembali sampai kembali ke titik awal yang terdapat pada paling sebelah kiri.

```
1  ALGORITMA ConvexHullAlgorithm(points, n)
2  //ConvexHullAlgorithm : membuat convexhull Jarvis March Algorthm
3  // Input : points adalah semua titik, n adalah jumlah titik
4  // Output : titik titik terluar hasil dari convex hull
5  hull ← zeros(0,0)
6  if n >= 3
7      l ← 1
8      for i ← 2 to n do
9          if points(i,2) < points(l,2)
10             l ← i
11     p ← l
12     while true
13         hull ← vertcat(hull, points(p,:))
14         q ← mod(p, n)+1
15         for i ← 1 to n do
16             answ ← Orientation(points(p,:), points(i,:), points(q,:))
17             if answ == 2
18                 q ← i
19         p ← q
20         if p == l
21             break
22 return hull
```

Gambar 4.7 *Pseudocode* Fungsi *ConvexHullAlgorithm*

Pada fungsi *Orientation* sendiri diimplementasikan oleh *pseudocode* yang dapat dilihat pada **Gambar 4.8**. Data masukan berupa *pp* adalah titik *minutiae* yang sudah terpilih sebelumnya, *qq* adalah titik *minutiae* kedua hasil iterasi, dan *rr* adalah titik *minutiae* ketiga hasil sementara yang akan dibandingkan. Sedangkan data keluaran fungsi *Orientation* adalah *f* adalah status 0 jika ketiga titik segaris, 1 jika 3 titik membentuk sudut searah jarum jam, dan 2 jika 3 titik membentuk sudut yang berkebalikan dengan arah jarum jam.

```
1  ALGORITMA Orientation(pp, qq, rr)
2   //LINGKARAN : orientation fungsi dari convex hull
3   // Input : terdiri dari point pp, qq, rr
4   // Output : hasil return apakah termasuk 0,1,2
5  val ← (qq(1,3) - pp(1,3)) * (rr(1,2)-qq(1,2)) -
   (qq(1,2)-pp(1,2)) * (rr(1,3) - qq(1,3))
6  if val == 0
7      f ← 0
8  if val>0
9      f ← 1
10 if val<0
11     f ← 2
12 return f
```

Gambar 4.8 *Pseudocode* Fungsi *Orientation*

4.2.2 Implementasi Transformasi Titik

Implementasi pada transformasi titik dimulai dari data hasil seleksi titik sebagai pusat untuk dijadikan *template/query*, data semua titik *minutiae* untuk titik tetangga. Selanjutnya akan dilakukan proses transformasi pra proses, lalu transformasi proses yang terdiri dari (rotasi, refleksi, dan translasi). Lalu hasil transformasi akan direpresentasikan ke dalam bentuk data (α,r,γ) untuk disimpan menjadi sebuah *template/query*.

Implementasi transformasi titik terdapat pada *pseudocode* fungsi *Transformasi* yang dapat dilihat pada **Gambar 4.9**. Dimulai dari data masukan *selectionPoint* sebagai titik *minutiae* yang terseleksi, *allPoint* sebagai semua titik *minutiae* untuk titik tetangga, dan *namafile* adalah nama *template/query* yang akan

disimpan. Pada baris 8 pembacaan *file template/query* dengan nama yang sesuai *namafile*. Selanjutnya pada baris 9 hingga 10, untuk semua titik pusat dari titik seleksi dilakukan fungsi *TransformasiPraProses*. Tujuannya adalah untuk membuat posisi baru dari semua titik tetangga terhadap titik seleksi yang menjadi titik pusat (0,0). Selain itu juga untuk memetakan seluruh titik tetangga kedalam sektor. Lalu pada baris 11 hingga 15, dilakukan proses transformasi menggunakan fungsi *TransformasiProses* untuk semua titik tetangganya terhadap titik pusat yang terseleksi lalu disimpan dalam bentuk data yang sudah direpresentasikan. Secara lebih jelasnya pada implementasi transformasi terbagi menjadi dua tahapan. Pertama adalah implementasi transformasi pra proses yang terdapat pada pada **Subbab 4.2.2.1**. Kedua implementasi transformasi dan representasi data yang terdapat pada **Subbab 4.2.2.2**.

```
1  ALGORITMA Transformasi(
   selectionPoint,allPoint,namafile )
2   //TRANSFORMASI : dimulai dari tranformasi pra proses,
   lalu transformasi yang
3   //terdiri dari rotasi, refleksi, dan translasi lalu
   disimpan
4   // Input : selectionPoint adalah array dari titik-
   titik yang terseleksi yang digunakan
5   // sebagai titik pusat, allPoint adalah semua titik,
   namafile adalah nama
6   // file titik untuk nama penyimpanan hasil
   tranformasi
7   // Output : hasil Template di simpan
8  fileID ← fopen(strcat(namafile,'.dat'), 'w')
9  for i ←1 to size(selectionPoint) do
10     matrix ←
   TransformasiPraProses(selectionPoint(i,:),allPoint)
11     for j ← 1 to length(matrix) do
12         if(selectionPoint(i,1) ~= matrix(j,1))
13             hasil ← TransformasiProses(matrix(j,:))
14             fprintf(fileID,'%f %f %f
   %f\n',selectionPoint(i,1),hasil)
15 fclose(fileID)
```

Gambar 4.9 *Pseudocode* Fungsi *Transformasi*

4.2.2.1 Implementasi Transformasi Pra Proses

Pada implementasi transformasi pra proses dilakukan reposisi titik *minutiae* dimana titik tetangga akan dirotasi terhadap titik pusat hasil seleksi titik sesuai orientasi titik pusat. Selain itu juga dilakukan pemetaan titik *minutiae* ke 32 sektor yang sudah ada. Transformasi pra proses diimplementasikan pada *pseudocode* fungsi *TransformasiPraProses* yang terdapat pada **Gambar 4.10**. Data masukan berupa *point* dan *matrix*. *Point* adalah titik yang terseleksi yang dijadikan titik pusat, sedangkan *matrix* adalah titik tetangga dari titik yang terseleksi. Sedangkan data keluaran fungsi *TransformasiPraProses* adalah *result* dalam bentuk *array* yang terdiri dari index *,* koordinat *x* baru, koordinat *y* baru, orientasi, dan index sektor yang dihasilkan.

Pada fungsi *TransformasiPraProses* baris 5 hingga 9, inisiasi untuk koordinat titik pusat dan orientasi. Baris 11 hingga 16, untuk setiap titik tetangga dilakukan rotasi terhadap orientasi dari titik pusat sehingga pada titik tetangga dihasilkan titik koordinat dan arah orientasi yang baru. Selain itu, fungsi *SektorMap* untuk menentukan sektor titik tetangga terhadap 32 sektor yang ada. Nilai kembalian pada fungsi *TransformasiPraProses* adalah *result* yang digunakan untuk *TransformasiProses* pada tahap berikutnya.

1	**ALGORITMA TransformasiPraProses(point,matrix)**
2	//TransformasiPraProses : Tahap tranformasi pra proses untuk memetakan titik yang lain pada titik pusat dan mengganti orientasi sesuai orientasi titik pusat
3	// Input : point adalah array titik tengah, matrix adalah array semua titik yang mau dipetakan lagi
4	// Output : result adalah array hasil proses tranformasi praproses yang terdiri dari index, titik baru x, titik baru y, orientasi baru dan index dari sektor map
5	cX ← point(1,2)
6	cY ← point(1,3)
7	cO ← point(1,4)
8	derajat ← cO/pi*180

```
titikPusat ← [cX;cY]
result ← []
for i ← for 1 to size(matrix) do
    titik ← [matrix(i,2);matrix(i,3)]
    derajatTitik ← matrix(i,4)/pi*180
    matrixRotasi ← [cosd(derajat),-sind(derajat);sind(derajat),cosd(derajat)]
    titikBaru ← matrixRotasi * double(titik-titikPusat)
    result ← [result ; [matrix(i,1),titikBaru(1),titikBaru(2),mod((derajatTitik-derajat),360)],SektorMap(titikBaru)]
return result
```

Gambar 4.10 *Pseudocode* Fungsi *TransformasiPraProses*

Pseudocode fungsi *SektorMap* terdapat pada **Gambar 4.11**. Data masukan berupa *titik* yang merupakan titik tetangga. Sedangkan data keluaran fungsi *SektorMap* adalah *sektor* berupa indeks sektor yang dihasilkan. Pada baris 7 hingga 20, dilakukan pemetaan titik koordinat(x,y) terhadap kuadran 1,2,3, atau 4. Baris 21, dilakukan pencarian arctan dari titik koordinat titik *minutiae*. Baris 22 hingga 91, dilakukan pemetaan sektor yang sesuai dengan keberadaan titik *minutiae*. Kembalian nilai *sektor* berupa bilangan 1 hingga 32 yang merupakan indeks sektor dari *titik* tersebut.

```
1  ALGORIMA SektorMap( titik )
2  //SEKTORMAP Summary of this function goes here
3  // Input : titik adalah titik yang akan dicari index sektor map
4  // Output : sektor adalah index hail dari sektor map
5  x ← titik(1)
6  y ← titik(2)
7  if(x>0 && y>0)
8      kuadran ← 1
9  elseif(x<0 && y>0)
10     kuadran ← 2
11 elseif(x<0 && y>0)
12     kuadran ← 3
13 elseif(x==0 && y<0)
14     kuadran ← 2
15 elseif(x>0 && y==0)
16     kuadran ← 1
```

```
elseif(x<0 && y==0)
    kuadran ← 3
else
    kuadran ← 4
sudut ← atand (double(y/x))
if(sudut>0)
    if(kuadran==1)
        if(sudut>=78.75)
            sektor ← 8
        elseif(sudut>=67.5)
            sektor ← 7
        elseif(sudut>=56.25)
            sektor ← 6
        elseif(sudut>=45)
            sektor ← 5
        elseif(sudut>=33.75)
            sektor ← 4
        elseif(sudut>=22.5)
            sektor ← 3
        elseif(sudut>=11.25)
            sektor ← 2
        else
            sektor ← 1
    else
            if(sudut>=78.75)
            sektor ← 24
        elseif(sudut>=67.5)
            sektor ← 23
        elseif(sudut>=56.25)
            sektor ← 22
        elseif(sudut>=45)
            sektor ← 21
        elseif(sudut>=33.75)
            sektor ← 20
        elseif(sudut>=22.5)
            sektor ← 19
        elseif(sudut>=11.25)
            sektor ← 18
        else
            sektor ← 17
else
    if(kuadran==2)
        if(sudut<=78.75)
            sektor ← 9
```

```
61          elseif(sudut<=67.5)
62              sektor ← 10
63          elseif(sudut<=56.25)
64              sektor ← 11
65          elseif(sudut<=45)
66              sektor ← 12
67          elseif(sudut<=33.75)
68              sektor ← 13
69          elseif(sudut<=22.5)
70              sektor ← 14
71          elseif(sudut<=11.25)
72              sektor ← 15
73          else
74              sektor ← 16
75      else
76          if(sudut<=78.75)
77              sektor ← 25
78          elseif(sudut<=67.5)
79              sektor ← 26
80          elseif(sudut<=56.25)
81              sektor ← 27
82          elseif(sudut<=45)
83              sektor ← 28
84          elseif(sudut<=33.75)
85              sektor ← 29
86          elseif(sudut<=22.5)
87              sektor ← 30
88          elseif(sudut<=11.25)
89              sektor ← 31
90          else
91              sektor ← 32
92  return sektor
```

Gambar 4.11 *Pseudocode* Fungsi *SektorMap*

4.2.2.2 Implementasi Transformasi dan Representasi Data

Pada implementasi transformasi dan representasi data dilakukan transformasi geometri (rotasi, refleksi dan translasi) dan dilakukan representasi data(sudut pusat, jarak dua titik, dan sudut titik *minutiae*). Implementasi ini terdapat pada *pseudocode* fungsi *TransformasiProses* yang dapat dilihat pada **Gambar 4.12**. Data

masukan berupa *titik_tetangga* yang merupakan titik tetangga. Sedangkan data keluaran fungsi *TransformasiProses* adalah *titikBaru* yang merupakan nilai representasi data.

Pada fungsi *TransformasiProses* baris 6 hingga 9 adalah inisiasi dari data titik_tetangga. Dilakukan pembacaan *key* dengan fungsi *ReadKey* pada baris 9. Selanjutnya sesuai *key* yang ada didapatkan nilai *far* sebagai pergeseran sektor untuk rotas, *keyX* dan *keyY* merupakan pergeseran koordinat *x* dan *y* untuk translasi. Pada baris 15 hingga 19, dilakukan inisiasi perhitungan matrix rotasi, refleksi, dan translasi. Pada baris 20 hingga 21, dilakukan perhitungan titik baru dan orientasi setelah dilakukan proses rotasi. Pada baris 22 hingga 30, dilakukan perhitungan titik baru dan orientasi setelah dilakukan proses refleksi. Dan baris 31, hanya dilakukan perhitungan titik baru setelah dilakukan translasi tanpa adanya orientasi baru pada saat proses translasi.

Selanjutnya baris 32 hingga 54 dilakukan perhitungan untuk representasi data yaitu *alfaTrans*(α) sebagai sudut pusat, *jarakTranslasi* (*r*) sebagai jarak dua titik dan *gama*(γ) sebagai sudut titik *minutiae*. Pada baris 55, *titikBaru* berisi representasi data sebagai nilai kembalian untuk fungsi *TransformasiProses*.

1	**ALGORITMA TransformasiProses(titik_tetangga)**
2	//TRANFORMASIPROSES adalah proses menstransformasi dengan rotasi, refleksi, dan translasi juga untuk merepresentasikan data
3	// Input : titik_tegangga adalah array dari titik tetangga yang akan
4	// ditransformasi
5	// Output : dihasilkan titikBaru hasil dari tranformasi untuk disimpan
6	pX ← titik_tetangga(1,2)
7	pY ← titik_tetangga(1,3)
8	pO ← titik_tetangga(1,4)
9	indexKey ← titik_tetangga(1,5)
10	key ← ReadKey()
11	far ← key(indexKey,1)
12	keyX ← key(indexKey,2)
13	keyY ← key(indexKey,3)
14	derajat ← double(11.25 * far)
15	matrixTranslasi ← [keyX;keyY]

```
16  matrixRotasi ← [cosd(derajat),-
    sind(derajat);sind(derajat),cosd(derajat)]
17  titik ← [pX;pY]
18  matrixRefleksiY ← [-1,0;0,1]
19  matrixRefleksiX ← [1,0;0,-1]
20  titikRotasi ← matrixRotasi * double(titik)
21  sudutRotasi ← mod(pO+derajat,360)
22  if(mod(indexKey,2)==0)
23      titikRefleksi ← matrixRefleksiX * titikRotasi
24      sudutRefleksi ← 360-sudutRotasi
25  elseif(mod(indexKey,2)==1)
26      titikRefleksi ← matrixRefleksiY * titikRotasi
27      if(sudutRotasi>180)
28      sudutRefleksi ← 180+(180-(sudutRotasi-180))
29      else
30      sudutRefleksi ← 180-sudutRotasi
31  tempTitikTranslasi ← titikRefleksi +
    double(matrixTranslasi)
32  c ← titikRefleksi(1)
33  d ← titikRefleksi(2)
34  alfaRef ← atand(double(d/c))
35  if(c>0 && d<0)
36      alfaRef ← alfaRef + 360
37  elseif(c<0)
38      alfaRef ← alfaRef + 180
39  elseif(c==0 && d<0)
40      alfaRef ← alfaRef + 270
41  e ← tempTitikTranslasi(1)
42  f ← tempTitikTranslasi(2)
43  alfaTrans ← atand(double(f/e))
44  if(e>0 && f<0)
45      alfaTrans ← alfaTrans + 360
46  elseif(e<0)
47      alfaTrans ← alfaTrans + 180
48  elseif(e==0 && f<0)
49      alfaTrans ← alfaTrans + 270
50  jarakTranslasi ← sqrt(e^2 + f^2)
51  if(0<c*d)
52      gama ← mod(sudutRefleksi,180)-mod(alfaTrans,180)
53  Else
54      gama ← mod(alfaTrans,180) - mod(sudutRefleksi,180)
55  titikBaru ← [alfaTrans, jarakTranslasi, gama]
56  return titikBaru
```

Gambar 4.12 *Pseudocode* Fungsi *TransformasiProses*

4.3 Implementasi Tahap Verifikasi Lokal dan Global

Implementasi verifikasi pada tugas akhir ini terdiri verifikasi lokal dan verifikasi global. Implementasi tahap verifikasi menggunakan *pseudocode* fungsi *Verifikasi* dapat dilihat pada **Gambar 4.13**. Data masukan berupa *namaquery* yang merupakan nama *file query, namatemplate* merupakan nama *file template, t1* adalah *threshold* sudut pusat, *t2* adalah *threshold* jarak dua titik, *t3* adalah *threshold* sudut titik *minutiae*, *t4* adalah *threshold* verifikasi lokal, dan *t*5 adalah *threshold* verifikasi global. Sedangkan data keluaran fungsi *Verifikasi* adalah *kepastian* antara 1 atau 0.

Pada fungsi *Verifikasi* pada baris 5 hingga 6, dilakukan pembacaan data *template* maupun *query* dengan menggunakan fungsi *ReadTemplate*. Pada baris 7 hingga 10, dilakukan pengelompokan data terhadap index *template* maupun *query*. Pada baris 11, dilakukan inisiasi *matrix_data* yang digunakan untuk menyimpan hasil dari verifikasi lokal. Baris 12 hingga 48, dilakukan verifikasi lokal. Pada baris 18 hingga 20, untuk setiap indeks yang sudah dikelompokkan antara *template* dan *query* dilakukan perhitungan selisih ketiga representasi data yaitu *selisihAlfa*, *selisihJarak*, dan *selisihGama*. Pada baris 21 hingga 24, dilakukan perbandingan apabila selisih ketiga representasi data tersebut lebih dari *threshold* (*t1,t2,t3*) maka *matrix_point* sebagai *adjacency matrix* verifikasi lokal akan menyimpan *selisihGama*. Jika tidak maka menyimpan nilai -1. Selanjutnya baris 26 hingga 48, dilakukan perhitungan dari *adjacency matrix* verifikasi lokal yang memenuhi untuk dibandingkan dengan *threshold* verifikasi lokal (*t4*). Apabila nilai memenui maka *matrix_data* sebagai *adjacency matrix* verifikasi global akan bernilai 1 jika tidak bernilai 0.

Verifikasi global diimplementasikan pada fungsi *verifikasi* baris 49 hingga 72. Pada baris 49 hingga 68, dilakukan perhitungan dari *adjacency matrix* verifikasi global yang memenuhi untuk dibandingkan dengan *threshold* verifikasi global (*t5*). Pada baris 69 hingga 72, dilakukan pengecekan terhadap *threshold* verifikasi

global, apabila memenuhi maka *kecocokan* akan bernilai 1 jika tidak *kecocokan* akan bernilai 0.

```
1  ALGORITMA Verifikasi( namequery, nametemplate, t1, t2, t3, t4, t5)
2  Verifikasi : verifikasi lokal dan global
3  // Input : namaquery adalah nama file query, namatemplate adalah nama file template, t1 adalah threshold alfa, t2 adalah threshold jarak, t3 adalah threshold gama,  t4 adalah threshold verifikasi lokal, t5 adalah threshold verifikasi global
4  // Output : kepastian cocok atau tidak cocok
5  template ← ReadTemplate( nametemplate )
6  query ← ReadTemplate( namequery )
7  counts_q ← histc(query(:,1),unique(query(:,1)))
8  query_data ← mat2cell(query(:,2:4),counts_q)
9  counts_t ← histc(template(:,1),unique(template(:,1)))
10 template_data ← mat2cell(template(:,2:4),counts_t)
11 matrix_data ← zeros(length(template_data),length(query_data))
12 for i ← 1 to length(template_data) do
13     for j ← 1 to length(query_data) do
14         count ← 0
15         matrix_point ← []
16         for a ← 1 to counts_t(1:1) do
17             for b ← 1 to counts_q(1:1) do
18                 selisihAlfa ← abs(template_data{i}(a,1)-query_data{j}(b,1))
19                 selisihJarak ← abs(template_data{i}(a,2)-query_data{j}(b,2))
20                 selisihGama ← abs(template_data{i}(a,3)-query_data{j}(b,3))
21                 if(selisihAlfa<t1 && selisihJarak<t2 && selisihGama<t3)
22                     matrix_point(a,b)← selisihGama
23                 else
24                     matrix_point(a,b)← -1
25         minimum ← min(counts_t(1:1),counts_q(1:1))
26         k ← 1
27         stop ← 0
28         count ← 0
29         while k<=minimum && stop ==0
30             kecil ← inf
31             for a ← 1 to counts_t(1:1) do
32                 for b ← 1 to counts_q(1:1) do
```

```
33                          if(matrix_point(a,b)~=-1)
34                              if(matrix_point(a,b)<kecil)
35                                  kecil ←  matrix_point(a,b)
36                                  baris ←  a
37                                  kolom ←  b
38                  if(kecil == inf)
39                      stop ←  1
40                  else
41                      count ←  count+1
42                      matrix_point(baris,:) ←  -1
43                      matrix_point(:,kolom) ←  -1
44                  k ←  k+1
45              if(count>t4*minimum)
46                  matrix_data(i,j) ←  1
47              else
48                  matrix_data(i,j) ←  0
49  k ←  1
50  stop ←  0
51  count ←  0
52  minimum ←
    min(length(template_data)/2,length(query_data)/2)
53  while k<=minimum && stop ==0
54  kecil ←  0
55      for i ←  1 to length(template_data) do
56          for j ←  1 to length(query_data) do
57              if(matrix_data(i,j)~=0)
58                  if(matrix_data(i,j)>kecil)
59                      kecil ←  matrix_data(i,j)
60                      baris ←  i
61                      kolom ←  j
62      if(kecil == 0)
63          stop ←  1
64      else
65          count ←  count+1
66          matrix_data(baris,:) ←  -1
67          matrix_data(:,kolom) ←  -1
68      k ←  k+1
69  if(count >= t5)
70      kepastian ←  1
71  else
72      kepastian ←  0
73  return kepastian
```

Gambar 4.13 *Pseudocode* Fungsi *Verifikasi*

(Halaman ini sengaja dikosongkan)

BAB V
HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi pada pengembangan proteksi sidik jari dengan metode *cancelable template* menggunakan *convex hull* dan proyeksi garis. Hasil uji coba didapatkan dari implementasi pada BAB 4IV dengan skenario yang berbeda. Bab ini berisikan pembahasan mengenai lingkungan pengujian, data pengujian, dan hasil pengujian.

5.1 Lingkungan Pengujian

Lingkungan pengujian pada uji coba permasalahan pengembangan proteksi sidik jari dengan metode *cancelable template* menggunakan *convex hull* dan proyeksi garis menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang terdapat pada **Tabel 5.1.**

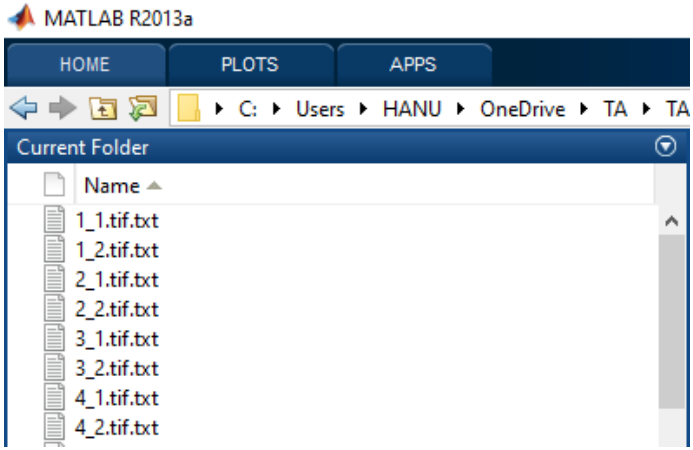
Tabel 5.1 Spesifikasi Lingkungan Pengujian

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i3-3240 CPU @ 3.40GHx (4CPUs), ~3.4 GHz
	Memori	4 GB 798.1 MHz DDR3
	VGA	AMD Radeon HD 8470 Graphics, Memory 2840 MB
Perangkat Lunak	Sistem Operasi	Windows 10 Pro 64 bit
	Perangkat Pengembang	MATLAB R2013a
	Bahasa Pemrograman	Matlab

Pengujian dilakukan dengan cara membuat kode yang telah dijelaskan pada implementasi BAB 4 lalu menjalankan programnya. Hasil dari pembuatan *template* & *query* akan disimpan dalam *file* berekstensi .dat. Untuk hasil dari verifikasi akan disimpan dalam *file* berbentuk csv. Pengujian dilakukan secara paralel untuk setiap skenario, yaitu dengan menjalankan MATLAB dalam mode *new window*. Jadi untuk satu skenario dijalankan dalam satu jendela MATLAB.

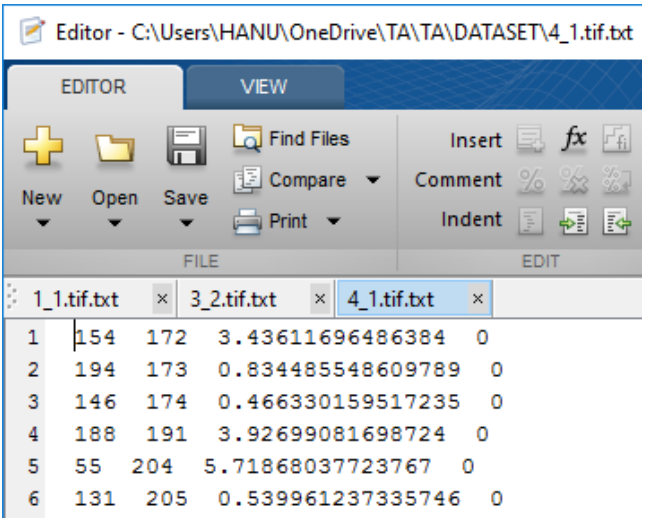
5.2 Data Pengujian

Subbab ini menjelaskan mengenai data yang digunakan pada uji coba. Data yang digunakan adalah *dataset* FVC2002DB2a dari *International competition for Fingerprint Verification Algorithms* [18]. Dalam *dataset* tersebut terdapat 100 pasang data sidik jari yang berbeda-beda. Karena terdapat 100 pasang diperlukan 10.000 verifikasi antara data *template* dan *query*. **Gambar 5.1** merupakan format *file dataset* yang diujicobakan yang disimpan dalam format ".txt".



Gambar 5.1 Format *File Dataset* Uji Coba

Untuk data *template* menggunakan nama *file* dengan nama "nomor_1.tif.txt", sedangkan untuk data *query* menggunakan nama *file* dengan nama "nomor_2.tif.txt". Di mana "nomor" adalah urutan sidik jari dari rentang 1 hingga 100. **Gambar 5.2** merupakan salah satu contoh isi data sidik jari. Isi data sidik jari berupa informasi titik *minutiae* oleh satu orang. Isi informasi tersebut adalah sumbu *x*, sumbu *y*, sudut orientasi titik *minutiae* dalam radian, serta tipe titik *minutiae* berupa nilai 0 atau 1 untuk menjelaskan *ridge ending* atau *ridge bifurcation*.

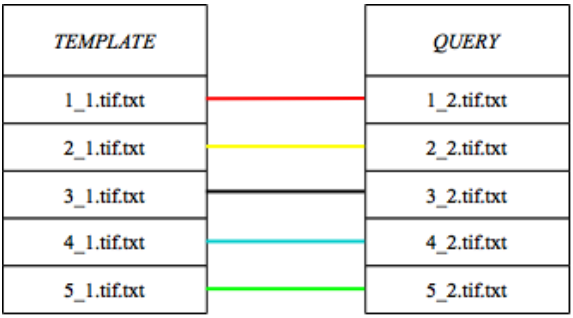


Gambar 5.2 Format Isi *Dataset* Uji Coba

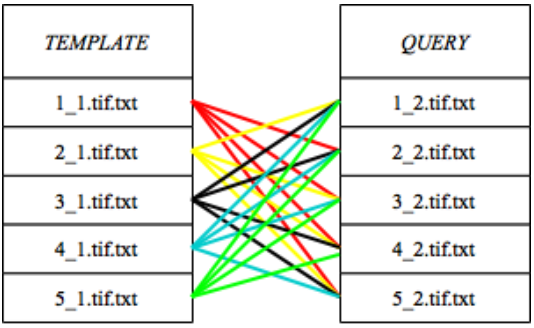
5.3 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini, dapat melihat keakuratan hasil dari metode yang digunakan dan dapat membandingkan skenario manakah yang memiliki hasil paling baik. Dalam uji coba diukur nilai TPR(*True Positif Rate*) dan FAR (*False Acceptance Rate*).

Pada TPR diukur dengan memasangkan data *template* dengan data *query* yang bersesuaian yang terdapat pada **Gambar 5.3**, misalkan *file* bernama "4_1.tif.txt" dicocokkan dengan *file* bernama "4_2.tif.txt". Sehingga terdapat pengujian sebanyak 100 kali. Nilai terbaik TPR yaitu 100 % jika 100 data yang dicocokkan ternyata cocok semua. Sedangkan FAR diukur dengan memasangkan data sebuah *template* dengan seluruh *query* selain pasangan *template* itu sendiri yang terdapat pada **Gambar 5.4**. Sehingga terdapat 99 kali untuk satu *template* dipasangkan ke *query* lainnya. Jika ada 100 *template* maka total verifikasi FAR ada 9900 kali pengujian.



Gambar 5.3 Pemasangan *Template* dan *Query* untuk Uji *True Positif Rate*(TPR)



Gambar 5.4 Pemasangan *Template* dan *Query* untuk Uji *False Acceptance Rate*(FAR)

Pada uji coba juga dibutuhkan kunci yang digunakan untuk melakukan transformasi. Kunci dibuat menggunakan fungsi random menggunakan **Kode Sumber 9.1** yang menghasilkan data seperti pada **Tabel 5.2.** Terdapat 32 baris yang berarti terdapat 32 sektor. Sedangkan kolom pertama merupakan kunci perpindahan sektor, kolom kedua kunci untuk translasi koordinat *x*, dan kolom ketiga kunci untuk translasi koordinat *y*.

Tabel 5.2 Kunci yang Digunakan Untuk Uji Coba

Perpindahan Sektor	Translasi X	Translasi Y
26	29	4
29	20	4
9	17	30
30	5	31
30	16	25
5	14	29
25	30	21
2	27	29
22	24	24
13	21	6
22	1	9
2	4	26
22	10	30
2	14	12
24	25	6
16	14	21
22	24	9
22	21	6

Perpindahan Sektor	Translasi X	Translasi Y
4	16	30
11	19	7
24	8	16
22	28	30
17	5	5
8	27	8
26	8	29
11	7	8
20	15	11
26	19	18
29	9	24
24	12	18
3	2	17
25	29	5

Terdapat empat skenario uji coba pada tugas akhir ini yang dijelaskan berikutnya pada **Subbab 5.3.1**, **Subbab 5.3.2**, **Subbab 5.3.3**, dan **Subbab 5.3.4**.

5.3.1 Skenario Uji Coba Pengaruh *Threshold*

Skenario pada pengaruh *threshold* dilakukan untuk pencarian nilai yang terbaik dan juga batasan dalam menjalankan program. Selain itu bisa menentukan *variable* dari representasi data yang sangat berpengaruh pada metode yang dijalankan. Pada metode yang akan diuji coba, terdapat lima *threshold*. Lima *threshold* tersebut yaitu alfa(α), jarak (r), gama(γ), verifikasi lokal, dan verifikasi global. Pada setiap kali uji coba dilakukan

perulangan untuk tiap uji coba, dimana ada penambahan nilai konstan pada sebuah *threshold* dan empat *threshold* lainnya menggunakan nilai tetap. Sehingga terdapat lima kali uji coba untuk masing-masing *threshold* agar mendapatkan parameter *threshold* terbaik. Kode sumber untuk menjalankan uji coba ini ditunjukkan pada **Kode Sumber 9.2**.

5.3.2 Skenario Uji Coba Pengaruh *Convex Hull*

Pada skenario ini, dicari pengaruh perbedaan area dari *convex hull* yang dilakukan. Sebelumnya terdapat tiga area yaitu T_{top} (area luar), T_{center} (area tengah), dan T_{bottom} (area dalam). Untuk mengetahui pengaruh *convex hull* dibuatkan beberapa kombinasi untuk dilakukan uji coba. Beberapa kombinasi uji coba pada pengaruh *convex hull* adalah sebagai berikut:

1. Menggunakan hasil *convex hull* pada area luar.
2. Menggunakan hasil *convex hull* pada area tengah.
3. Menggunakan hasil *convex hull* pada area dalam.
4. Menggunakan hasil *convex hull* pada area luar dan area tengah.
5. Menggunakan hasil *convex hull* pada area luar dan area dalam.
6. Menggunakan hasil *convex hull* pada area tengah dan area dalam.
7. Menggunakan hasil *convex hull* pada area luar dan area tengah dan area dalam.
8. Menggunakan seluruh data sidik jari atau tidak menggunakan *convex hull* dalam proses seleksi titik.

Dalam uji coba penulis menggunakan nilai *threshold* yang disamaratakan yaitu nilai T1 sebesar 18, T2 sebesar 20, T3 sebesar 14, T4 sebesar 0.38, dan T5 sebesar 4. Uji coba pengaruh *convex hull* ini menggunakan **Kode Sumber 9.3** untuk mengontrol jalannya uji coba.

5.3.3 Skenario Uji Coba terhadap Waktu *Running*

Pada skenario ini, dilakukan uji coba untuk waktu *running* yang dibutuhkan antara pengujian tanpa adanya seleksi titik dengan pengujian dengan seleksi titik yang menggunakan *threshold area* dan *convex hull* yang diusulkan. Waktu running yang diujikan pada waktu pembuatan *template/query* dan pada waktu verifikasi antara *template* dan *query*. Selain itu, perbedaan jumlah titik yang digunakan sebagai pusat dalam membentuk *template* juga dihitung rata-ratanya. Dari uji skenario waktu dan hasil FAR & TPR akan menghasilkan performa pengujian kode program. Kode sumber untuk menjalankan uji coba ini ditunjukkan pada **Kode Sumber 9.4**.

5.3.4 Skenario Uji Coba Pencarian EER

Uji skenario yang keempat yaitu pencarian EER (*Equal Error Rate*). *Equal Error Rate* merupakan nilai yang sering digunakan dalam pengukuran dari metode verifikasi sidik jari. Untuk mendapatkan EER dilakukan *plotting* terhadap nilai FAR dengan nilai FRR (*False Rejection Rate*). FRR didapatkan dari mengurangkan 100 dengan nilai TPR yang dihasilkan. Skenario uji coba hampir sama seperti uji coba terhadap pengaruh *threshold*, dimana empat nilai *threshold* dijadikan konstan dan satu *threshold* nilainya berubah-ubah. Pada skenario uji coba ini, menghasilkan grafik EER yang sama dengan jumlah uji skenario pengaruh *threshold* karena menggunakan data yang sudah ada.

5.4 Uji Coba

Uji coba yang dilakukan pada tugas akhir ini sesuai skenario yang disebutkan pada **Subbab 5.3** yang dijelaskan sebagai berikut :

5.4.1 Uji Coba Pengaruh *Threshold*

Uji coba pengaruh *threshold* dilakukan untuk data yang di transformasi untuk lima *threshold* yaitu alfa(T1), jarak(T2), gama(T3), verifikasi lokal(T4), dan verifikasi global(T5). Hasil dari uji coba ditunjukkan pada **Tabel 5.3**, **Tabel 5.4**, **Tabel 5.5**, **Tabel 5.6**, dan **Tabel 5.7.**

.

Tabel 5.3 Data Perubahan *Threshold* T1

T1	T2	T3	T4	T5	TPR(%)	FAR(%)
16	20	14	0,38	4	90	2,97
18	20	14	0,38	4	93	3,84
20	20	14	0,38	4	93	4,88
22	20	14	0,38	4	93	6,33
24	20	14	0,38	4	94	7,71

Tabel 5.4 Data Perubahan *Threshold* T2

T1	T2	T3	T4	T5	TPR(%)	FAR(%)
18	18	14	0,38	4	90	2,36
18	19	14	0,38	4	91	3,13
18	20	14	0,38	4	93	3,84
18	21	14	0,38	4	94	4,69
18	22	14	0,38	4	94	5,76
18	23	14	0,38	4	95	6,91

Tabel 5.5 Data Perubahan *Threshold* T3

T1	T2	T3	T4	T5	TPR(%)	FAR(%)
18	20	12	0,38	4	88	1,86
18	20	13	0,38	4	89	2,90
18	20	14	0,38	4	93	3,84
18	20	15	0,38	4	93	5,00
18	20	16	0,38	4	93	6,77
18	20	17	0,38	4	96	8,78

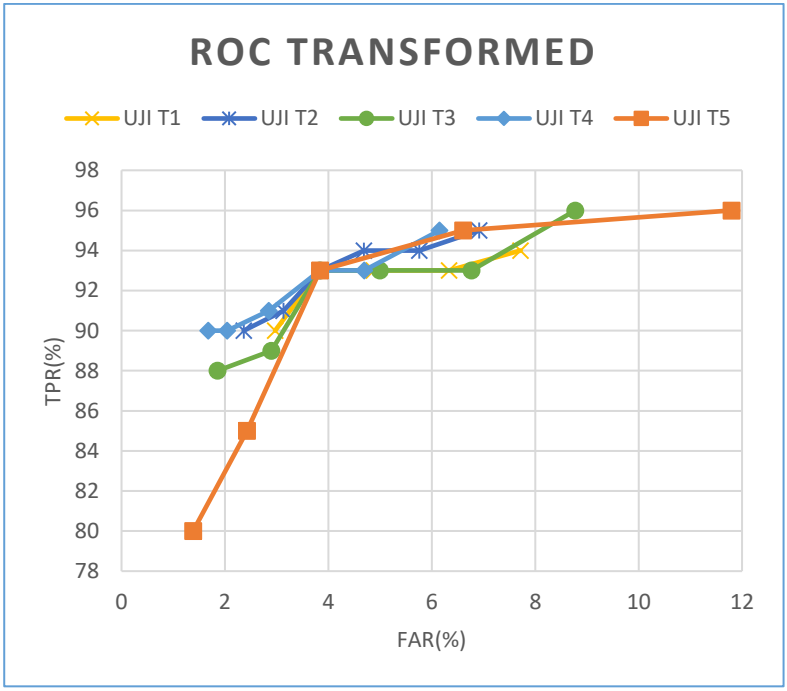
Tabel 5.6 Data Perubahan *Threshold* T4

T1	T2	T3	T4	T5	TPR(%)	FAR(%)
18	20	14	0,36	4	95	6,15
18	20	14	0,37	4	93	4,70
18	20	14	0,38	4	93	3,89
18	20	14	0,39	4	91	2,85
18	20	14	0,40	4	90	2,04
18	20	14	0,41	4	90	1,68

Tabel 5.7 Data Perubahan *Threshold* T5

T1	T2	T3	T4	T5	TPR(%)	FAR(%)
18	20	14	0,38	2	95	6,15
18	20	14	0,38	3	93	4,70
18	20	14	0,38	4	93	3,84
18	20	14	0,38	5	91	2,85
18	20	14	0,38	6	90	2,04

Dari hasil uji coba pada **Tabel 5.3**, **Tabel 5.4**, dan **Tabel 5.5**, merupakan uji pengaruh *threshold* pada representasi data alfa(α), jarak (r), dan gama(γ). Ketiga representasi data tersebut dibuat menggunakan proyeksi garis untuk mendapatkan nilainya. Selain itu tiga nilai representasi data[6] mendapatkan hasil EER yang lebih baik daripada menggunakan dua nilai representasi data [7]. Pada hasil coba pengaruh *threshold* dapat diketahui bahwa semakin besar nilai *threshold* pada representasi data maka semakin besar juga nilai TPR dan FAR. Sedangkan pada **Tabel 5.6** dan **Tabel 5.7,** merupakan uji pengaruh *threshold* pada verifikasi lokal dan global. Semakin besar nilai *threshold* pada verifikasi maka semakin kecil nilai TPR dan FAR. Dari hasil uji coba pengaruh *threshold* dibuat sebuah grafik yang merepresentasikan ROC yang ditunjukkan pada **Gambar 5.5**.



Gambar 5.5 Grafik ROC *Transformed*

5.4.2 Uji Coba Pengaruh *Convex Hull*

Pada uji coba pengaruh *convex hull*, dilakukan uji coba melakukan semua kombinasi uji coba terhadap setiap perubahan nilai TPR dan FAR. Uji coba ini untuk mengetahui pengaruh perbedaan jumlah pengambilan titik saat melakukan proses seleksi titik. Uji coba ini menghasilkan data yang ditunjukkan pada **Tabel 5.8**.

Tabel 5.8 Hasil Uji Coba Pengaruh *Convex Hull*

No	Tahapan Convex Hull	TPR (%)	FAR (%)	Rata-rata titik yang terseleksi
1	Area luar	39	0,26	9
2	Area tengah	25	0,17	8
3	Area dalam	Tidak dapat dihitung		
4	Area luar dan tengah	78	1,94	15
5	Area luar dan dalam	66	1,06	14
6	Area tengah dan dalam	73	1,17	12
7	Area luar, tengah, dan dalam	93	3,84	20
8	Semua titik tanpa *convex hull*	98	23,67	40

Dari hasil uji coba bisa kita lihat bahwa ada pengaruh antara tahapan *convex hull* nilai TPR dan FAR serta rata-rata titik yang terseleksi untuk satu *template*. Semakin banyak tahapan seleksi semakin banyak juga nilai TPR dan FAR.

5.4.3 Uji Coba terhadap Waktu *Running*

Pada uji coba terhadap waktu *running* dilakukan untuk mengetahui perbedaan waktu pada waktu membuat *template* dan melakukan verifikasi antara *template* dan *query*. Uji coba ini untuk membandingkan antara uji coba menggunakan seleksi titik (*threshold area* dan *convex hull* menggunakan tiga area) atau metode yang penulis usulkan dengan tanpa seleksi titik (semua titik). Uji coba ini meliputi waktu *running* yang dapat dilihat pada **Tabel 5.9** untuk pembuatan *template* dan **Tabel 5.10** untuk proses verifikasi.

Tabel 5.9 Waktu *Running* Pembuatan *Template*

Percobaan	Dengan Seleksi (detik)	Tanpa Seleksi (detik)
1	0,64	1,36
2	0,63	1,35
3	0,64	1,36
4	0,64	1,35
5	0,63	1,33
6	0,62	1,29
7	0,62	1,30
8	0,62	1,32
9	0,62	1,33
10	0,62	1,32
Rata-rata	**0,63**	**1,33**

Tabel 5.10 Waktu *Running* Proses Verifikasi

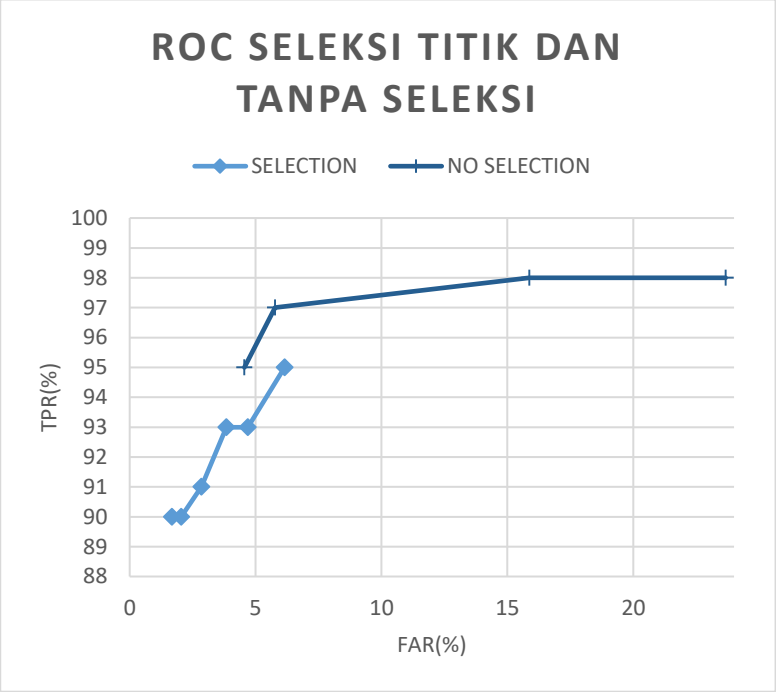
Percobaan	Dengan Seleksi (detik)	Tanpa Seleksi (detik)
1	10,06	49,07
2	10,48	48,89
3	10,04	48,61
Rata-rata	**10,19**	**48,86**

Tabel 5.11 Data Perubahan Tanpa Seleksi Titik

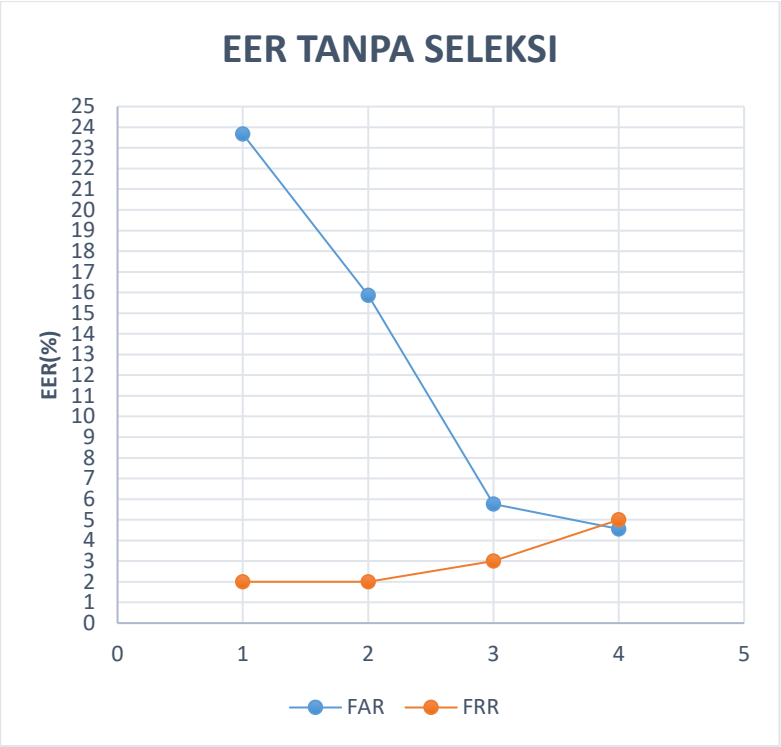
T1	T2	T3	T4	T5	TPR(%)	FAR(%)
18	20	14	0,38	4	98	23,67
18	20	14	0,40	4	98	15,87
18	20	14	0,45	4	97	5,77
18	20	14	0,46	4	95	4,56

Pada **Tabel 5.9** dapat dilihat percobaan waktu *running* untuk pembuatan *template* dilakukan sepuluh kali percobaan lalu dicari rata-rata untuk masing-masing proses yang dengan seleksi maupun tanpa seleksi. Hasil menunjukkan bahwa proses seleksi titik lebih

cepat dua kali lipat dari pada tanpa seleksi titik. Sedangkan pada **Tabel 5.10** dapat dilihat percobaan waktu *running* verifikasi dilakukan tiga kali percobaan dan dihasilkan bahwa proses verifikasi dengan seleksi titik lebih cepat lima kali lipat dari pada tanpa seleksi titik. Hasil itu berbanding terbalik dengan akurasi yang dibuktikan dengan grafik ROC pada **Gambar 5.6**. Grafik ROC dibentuk dari **Tabel 5.11** yang merupakan uji coba ketika tanpa seleksi titik dengan **Tabel 5.6** yang merupakan uji coba ketika menggunakan seleksi titik dengan variasi T4. Selain itu dilakukan pencarian EER dari **Tabel 5.11** yang ditunjukkan pada gambar **Gambar 5.7** menghasilkan nilai EER sebesar 4,72%. Untuk selisih antara EER seleksi titik yang dihitung pada **Subbab 5.4.4** dengan tanpa seleksi titik sebesar 0,95%.



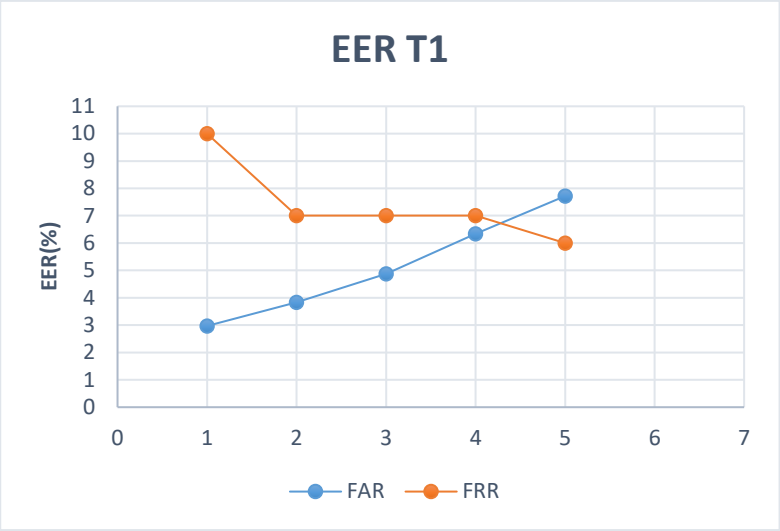
Gambar 5.6 ROC Seleksi Titik dan Tanpa Seleksi Titik



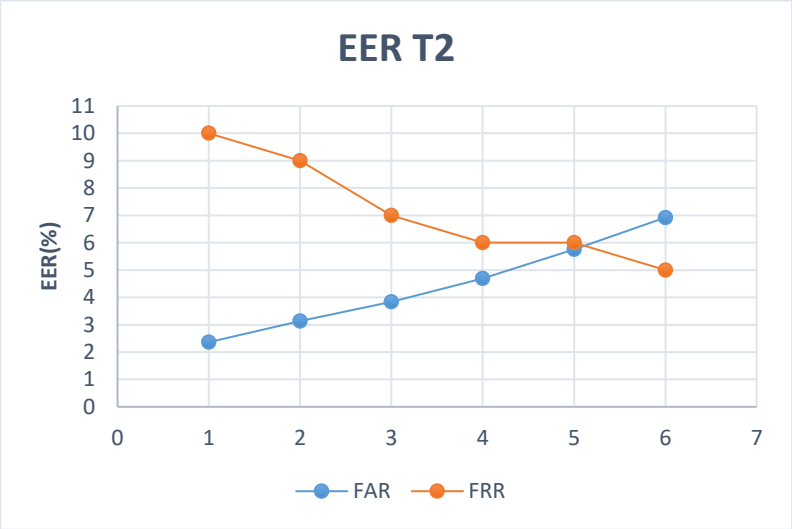
Gambar 5.7 ERR Tanpa Seleksi Titik

5.4.4 Uji Coba Pencarian ERR

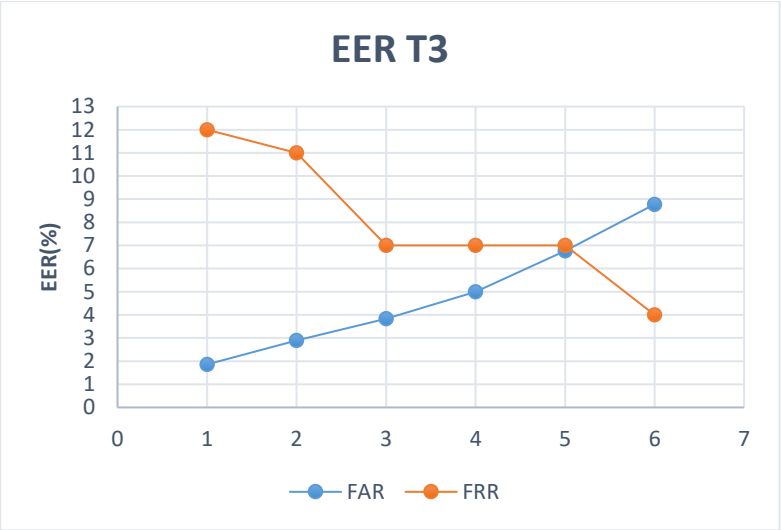
Pada pencarian ERR dilakukan pembuatan grafik EER dari hasil pengujian pencarian *threshold* dengan data yang tertera pada **Tabel 5.3**, **Tabel 5.4**, **Tabel 5.5**, **Tabel 5.6**, dan **Tabel 5.7**. Grafik EER yang dihasilkan ditunjukkan pada **Gambar 5.8**, **Gambar 5.9**, **Gambar 5.10**, **Gambar 5.11**, dan **Gambar 5.12**.



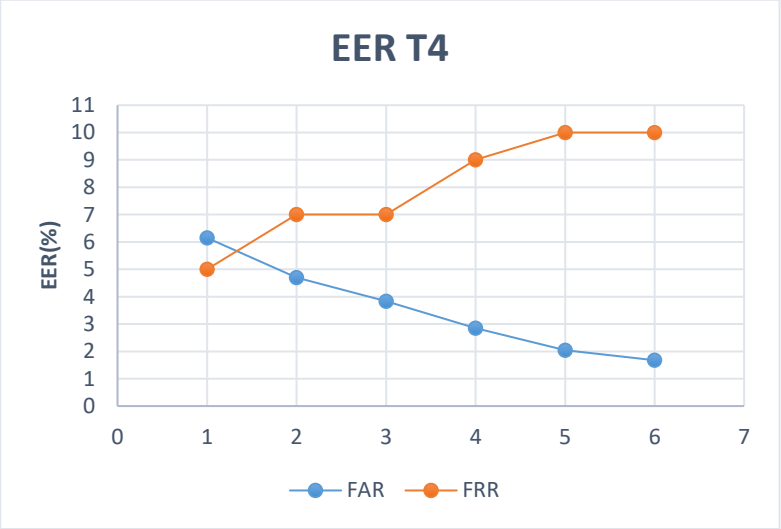
Gambar 5.8 Grafik EER T1 *Transformed*



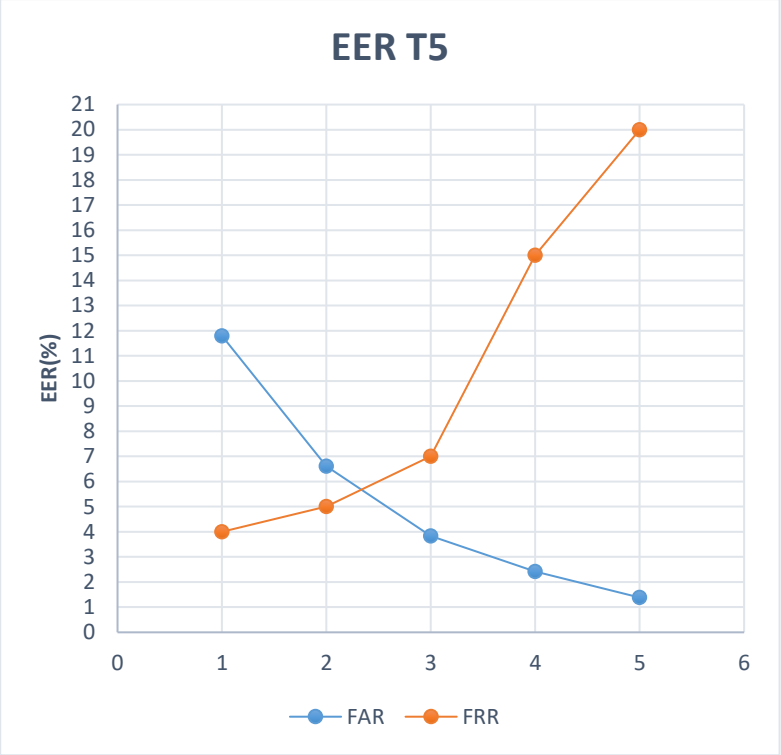
Gambar 5.9 Grafik EER T2 *Transformed*



Gambar 5.10 Grafik EER T3 *Transformed*



Gambar 5.11 Grafik EER T4 *Transformed*



Gambar 5.12 Grafik EER T5 *Transformed*

Tabel 5.12 Hasil Pencarian EER

EER T1(%)	EER T2(%)	EER T3(%)	EER T4(%)	EER T5(%)
6.86	5,89	6,86	5,67	5,67

Dari hasil masing-masing grafik EER dibuat tabel EER untuk masing-masing *threshold* yang terdapat pada **Tabel 5.12**. Nilai EER terkecil yang didapat dari proses transformasi adalah 5,67 %. Semakin kecil EER yang dihasilkan, maka akan semakin baik keakuratan proses verifikasinya.

5.5 Evaluasi

Pada subbab ini akan dijelaskan hasil dari serangkaian uji coba yang dilakukan dan kendala yang dihadapi selama proses pengerjaan. Evaluasi yang dilakukan sesuai dengan skenario uji coba yang dilakukan dan juga terdapat evaluasi analisis keamanan.

5.5.1 Evaluasi Uji Coba Pengaruh *Threshold*

Pada uji coba pengaruh *threshold* telah didapatkan bahwa semakin besar nilai *threshold* pada representasi data, maka besar juga nilai TPR dan FAR. Hal ini dikarenakan *threshold* dijadikan batas maksimal. Ketika rentang pembatas pada representasi data dibuat besar maka kesempatan data *template* dan *query* untuk cocok semakin besar juga. Representasi data yang digunakan yaitu *threshold* T1 untuk alfa(α), *threshold* T2 untuk jarak (r), dan *threshold* T3 untuk gama(γ). Evaluasi yang dilakukan terdapat pada nilai jangkauan representasi data yang mempengaruhi perbedaan akurasi pada nilai TPR dan FAR. Untuk alfa(α) dari data *template* yang ada jangkauannya antara 0 hingga 360, lalu jarak(r) jangkauannya antara 0 hingga 400, dan gama(γ) jangkauannya antara -180 hingga 180. Semakin banyak jangkauan data yang digunakan maka akan mengurangi pencocokan palsu.

Pada uji coba pengaruh *threshold* pada verifikasi lokal dan global semakin besar nilai *threshold* maka semakin kecil nilai TPR dan FAR. Hal ini dikarenakan nilai *threshold* ini digunakan sebagai batas minimal *template* dan *query* bisa dikatakan cocok. Kedua nilai *threshold* T4 untuk batas minimal verifikasi lokal dan T5 untuk batas minimal verifikasi global. Pada *threshold* T4 menggunakan nilai pecahan atau dibawah nilai satu karena menggunakan nilai persenan dari minimal total data antara *template* dan *query*. Pada *threshold* T5 menggunakan nilai *fix* umumnya titik *minutiae* secara global bisa dikatakan cocok jika ada empat nilai data yang cocok. Ini terbukti dari uji *threshold* T5, nilai empat mendapatkan hasil terbaik untuk TPR dan FAR.

5.5.2 Evaluasi Uji Coba Pengaruh *Convex Hull*

Pada uji coba pengaruh *convex hull* telah didapatkan adanya pengaruh dari tahapan *convex hull* untuk nilai TPR, FAR serta rata-rata titik yang terseleksi untuk satu *template*. Evaluasi dilakukan pada pada tahapan *convex hull* untuk area dalam tidak bisa dilakukan penghitungan karena ada beberapa dataset baik *template* maupun *query* tidak mendapatkan titik sama sekali sehingga tidak ada yang diverifikasi. Pada uji pengaruh *convex hull* pada area luar, area tengah, dan area dalam yang menjadi metode yang diusulkan mendapatkan nilai normal dengan rentang yang tidak begitu jauh antara TPR dan FAR. Tetapi ada beberapa titik yang sama-sama terseleksi pada waktu *convex hull* dengan area yang berbeda. Dalam hal ini akan diambil satu titik *minutiae* saja ketika ada yang *redundant*.

Rata-rata titik yang terseleksi dihitung dengan cara menghitung rata-rata dari minimal titik yang terseleksi antara data *template* dengan *query*. Semakin luar area yang digunakan semakin banyak titik yang terseleksi. Semakin banyak tahapan *convex hull* pada beberapa area yang dilakukan juga semakin besar nilai TPR dan FAR. Namun dengan banyaknya titik yang terseleksi juga semakin lama proses verifikasinya.

5.5.3 Evaluasi Uji Coba terhadap Waktu *Running*

Pada uji coba pengaruh waktu *running* antara seleksi titik dan tanpa seleksi titik mendapatkan hasil 2:1 untuk pembuatan *template* dan 5:1 untuk proses verifikasi. Waktu *running* ini dipengaruhi oleh jumlah titik *minutiae* yang terseleksi yang digunakan sebagai pembuatan *template*. Semakin banyak jumlah titik *minutiae* yang terseleksi maka semakin banyak *template* yang dibuat dan juga menyebabkan semakin banyak juga waktu *running* yang diperlukan. Selain itu, juga berpengaruh terhadap akurasi nilai EER. Waktu *running* berbanding terbalik dengan akurasi nilai EER yang dihasilkan. Namun selisih nilai EER tidak terlalu jauh

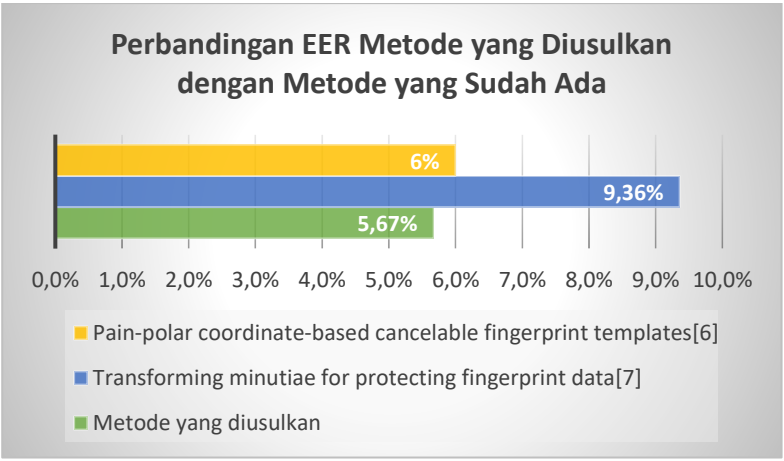
dibandingkan dengan selisih kecepatan waktu *running* yang diperlukan.

5.5.4 Evaluasi Uji Coba Pencarian EER

Pada uji coba uji coba pencarian ERR didapatkan nilai EER terkecil yaitu sebesar 5,67%. Nilai EER sebesar 5,67% ini didapatkan dari uji coba *threshold* T4 dan T5. EER didapatkan dari *plotting* antara nilai FAR dengan nilai FRR. Kendala dalam pencarian EER adalah diharuskan grafik antara nilai FAR dan FRR harus berpotongan, dimana perpotongan itu merupakan nilai EER yang dihasilkan. Nilai EER digunakan untuk menghitung akurasi pada metode verifikasi sidik jari. Semakin besar nilai EER maka hasil akurasi dari metode verifikasi sidik jari yang diproses semakin buruk. Dari hasil EER yang didapatkan dapat digunakan sebagai pembanding dengan metode-metode lain yang sudah pernah dilakukan. Hasil metode yang diusulkan lebih baik daripada metode sebelum-sebelumnya yang ditunjukkan pada **Tabel 5.13** dengan grafik yang dapat dilihat pada **Gambar 5.13**. Metode yang diusulkan mendapatkan hasil 5,67% yang mempunyai selisih 3,69% dari referensi utama[7] pembuatan tugas akhir ini. Hasil ini membuktikan beberapa tahapan pada *cancelable template* yang dilakukan variasi pada **Tabel 9.1** berhasil dilakukan dan mendapatkan hasil yang optimal.

Tabel 5.13 Perbandingan EER Metode yang Diusulkan dengan Metode yang Sudah Ada

Metode yang diusulkan	**Transforming minutiae for protecting fingerprint data**[7]	**Pair-polar coordinate-based cancelable fingerprint templates**[6]
5,67 %	9,36%	6 %



Gambar 5.13 Grafik Perbandingan EER Metode yang Diusulkan dengan Metode yang Sudah Ada

5.5.5 Evaluasi Analisis Keamanan

Analisis keamanan dilakukan dengan kemungkinan *template* data yang dibuat bisa dikembalikan ke nilai aslinya untuk mendapat data sidik jari asli. Data *template* yang disimpan adalah data hasil dari representasi data yang direpresentasikan dalam bentuk *tuple* (α, r, γ) dimana α merupakan sudut pusat, r merupakan jarak titik, dan γ merupakan sudut titik *minutiae*. Untuk mencari nilai α maka harus tahu nilai α sebelum dilakukan translasi, refleksi, dan rotasi dengan *key*. Meskipun *key* sudah didapat namun informasi sektor telah dihilangkan, dimana sektor yang sebanyak 32 bagian dibutuhkan untuk melakukan translasi, refleksi, dan rotasi. Sehingga satu satunya cara yang dapat digunakan adalah cara *brute force* dengan mencari segala mungkin bentuk kombinasi yang digunakan.

Dimulai dengan nilai α anggap saja ketelitian yang digunakan sebanyak 3 angka di belakang koma maka terdapat 360.000 kemungkinan. Selanjutnya pada nilai r anggap sama menggunakan ketelitian 3 angka di belakang koma serta rentang

terjauh dari data yang ada adalah 400 maka terdapat 400.000 kemungkinan. Untuk nilai γ diberikan keakuratan 3 angka di belakang koma menghasilkan kemungkinan sebanyak 360.000. Sehingga untuk mencari informasi titik *minutiae* memerlukan kemungkinan yang banyak dengan mengalikan kemungkinan *α,* kemungkinan *r*, dan kemungkinan γ dengan besar kemungkinan sebanyak 5,184 x 10^{16}. Itu belum termasuk menghitung komputasi operasi dasar ataupun instruksi lainnya yang menambah lamanya komputasi.

Sedangkan untuk *template* yang tanpa melakukan transformasi, jumlah kemungkinan yang dihasilkan lebih sedikit. Jika *α* sama dengan nilai *α* sebelum ditransformasi cukup dilakukan kemungkinan yang terjadi pada nilai *α*, maka secara otomatis akan mendapatkan nilai *r* dan juga nilai γ yang didapat dari nilai *α* dan nilai φ. Sehingga dengan ketelitian tiga angka di belakang koma maka memerlukan 360.000 kemungkinan.

Sehingga untuk perbedaan antara yang ditransformasi dengan yang tidak dengan cara *brute force* yaitu sebesar 5,184 x 10^{16} berbanding 3,6 x 10^{5}. Meskipun kemungkinan terburuk dapat tercapai namun pencuri tidak bisa mendapatkan data sidik jari secara asli karena beberapa titik dihilangkan pada waktu proses seleksi titik menggunakan *threshold area* dan *convex hull* sebelum melakukan transformasi. Sehingga metode pada *cancelable template* yang diusulkan bersifat *irreversible* karena *template* yang dibuat tidak bisa dikembalikan lagi menjadi data titik *minutiae* semula.

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan metode.

7.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba pengembangan proteksi sidik jari dengan metode *cancelable template* menggunakan *convex hull* dan garis proyeksi adalah sebagai berikut:

1. Pengembangan untuk melakukan variasi terhadap setiap tahapan mulai dari seleksi titik, transformasi, representasi data, dan proses verifikasi berhasil dilakukan dengan hasil EER sebesar 5,69%.
2. Seleksi titik menggunakan *threshold area* dan *convex hull* pada tiga area dapat mempercepat proses pembuatan *template* dua kali lebih cepat dan pada proses verifikasi lima kali lebih cepat dibandingkan dengan tanpa seleksi titik. Namun akurasi berkurang dengan selisih nilai EER sebesar 0,95%.
3. Metode seleksi titik menggunakan *convex hull* dan representasi data menggunakan proyeksi garis dengan memperbanyak *variable* yang tidak *redundant* dapat menambah akurasi dengan selisih 3,69% untuk nilai EER dari referensi utama[7].
4. Keamanan pada metode *cancelable template* yang diusulkan bersifat *irreversible* dan hanya bisa dibobol dengan cara *bruteforce* dengan kemungkinan yang dihasilkan sebesar 5,184 x 10^{16} kemungkinan untuk ketelitian tiga angka di belakang koma.

7.2 Saran

Saran yang diberikan terkait pengembangan pada tugas akhir ini adalah sebagai berikut :

1. Penentuan mekanisme pada tahap seleksi titik dengan area yang bisa mewakili semua titik *minutiae*.
2. Pemilihan *variable* representasi data dengan jangkauan nilai yang besar sehingga tidak terjadi pencocokan palsu.

DAFTAR PUSTAKA

[1] T. Ahmad and J. Hu, "Generating cancelable biometric templates using a projection line," in *2010 11th International Conference on Control Automation Robotics Vision*, 2010, pp. 7–12.

[2] T. Ahmad, D. S. Pambudi, and T. Usagawa, "Improving the performance of projection-based cancelable fingerprint template method," in *2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, 2015, pp. 84–88.

[3] *Handbook of Fingerprint Recognition | Davide Maltoni | Springer*. .

[4] A. K. Jain, K. Nandakumar, and A. Nagar, "Biometric template security," *EURASIP J. Adv. Signal Process.*, vol. 2008, p. 113, 2008.

[5] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle, "Generating Cancelable Fingerprint Templates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 561–572, Apr. 2007.

[6] T. Ahmad, J. Hu, and S. Wang, "Pair-polar coordinate-based cancelable fingerprint templates," *Pattern Recognit.*, vol. 44, no. 10–11, pp. 2555–2564, Oct. 2011.

[7] T. Ahmad, H. Markoni, W. Wibisono, and R. M. I, "Transforming minutiae for protecting fingerprint data," in *2015 International Symposium on Technology Management and Emerging Technologies (ISTMET)*, 2015, pp. 213–217.

[8] L. Hong, Y. Wan, and A. Jain, "Fingerprint image enhancement: algorithm and performance evaluation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 777–789, Agustus 1998.

[9] H. Yang, X. Jiang, and A. C. Kot, "Generating secure cancelable fingerprint templates using local and global features," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 645–649.

[10] A. M. Andrew, “Another efficient algorithm for convex hulls in two dimensions,” *Inf. Process. Lett.*, vol. 9, no. 5, pp. 216–219, Dec. 1979.
[11] C. Harrison, “Chris Harrison | ConvexHull,” *An Investigation of Graham’s Scan and Jarvis’ March*. [Online]. Available: http://www.chrisharrison.net/index.php/Research/ConvexHull. [Accessed: 25-Apr-2017].
[12] R. A. Jarvis, “On the identification of the convex hull of a finite set of points in the plane,” *Inf. Process. Lett.*, vol. 2, no. 1, pp. 18–21, Mar. 1973.
[13] T. Koetsier and L. Bergmans, *Mathematics and the Divine: A Historical Study*. Elsevier, 2004.
[14] E. W. Weisstein, “Trigonometry.” [Online]. Available: http://mathworld.wolfram.com/Trigonometry.html. [Accessed: 16-May-2017].
[15] J. E. Gentle, *Matrix Algebra: Theory, Computations, and Applications in Statistics*. Springer Science & Business Media, 2007.
[16] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Springer Science & Business Media, 2009.
[17] X. Yao, D. Gong, and Y. Gu, “Mathematic model of node matching based on adjacency matrix and evolutionary solutions,” *Phys. Stat. Mech. Its Appl.*, vol. 416, pp. 354–360, Dec. 2014.
[18] “FVC2002 - Second International Fingerprint Verification Competition.” [Online]. Available: http://bias.csr.unibo.it/fvc2002/databases.asp. [Accessed: 02-May-2017].
[19] “MATLAB - MathWorks.” [Online]. Available: https://www.mathworks.com/products/matlab.html. [Accessed: 09-May-2017].
[20] L. D. Singh, P. Das, and N. Kar, “A pre-processing algorithm for faster convex hull computation,” in *Confluence 2013: The Next Generation Information Technology Summit (4th International Conference)*, 2013, pp. 413–418.

LAMPIRAN

Lampiran 1. Kumpulan Kode Sumber yang Digunakan

```
1  function hasil =  generateKey();
2  tes  =  randi ( [1 31], 32 , 3 ); %random 1-31
3  fileID  =  fopen('Key.dat', 'w'); %save to Key.dat
4  fprintf(fileID,'%d %d %d\n',tes);
5  fclose(fileID);
6  end
```

Kode Sumber 9.1 *Generate Key*

```
1  function hasil =  FindThreshold()
2  for t1 = [t1_awal:t1_increment:t1_akhir]
3      for t2 = [t2_awal:t2_increment:t2_akhir]
4          for t3 = [t3_awal:t3_increment:t3_akhir]
5              for t4 = [t4_awal:t4_increment:t4_akhir]
6                  for t5 =
   [t5_awal:t5_increment:t5_akhir]
7                      skenario='DATA/SELECTION/';
8                      hasil =
   test_skenario1(18,20,14,0.38,4,skenario);
9                      GAR=hasil(1)/100*100;
10                     FAR=hasil(2)/9900*100;
11                     fileID =
   fopen(strcat(skenario,'FIND_THRESHOLD.csv'), 'at');
12
   fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f\n',t1,t2,t3,t4,t5,
   GAR,FAR);
13                     fclose(fileID);
14                 end
15             end
16         end
17     end
18 end
19 end
```

Kode Sumber 9.2 Uji Coba Pengaruh *Threshold*

```
1  function hasil =  UjiConvexHull()
2  hasil = test_skenario2(18,20,14,0.38,4,'DATA/TOP/');
3  GAR=hasil(1)/100*100;
4  FAR=hasil(2)/9900*100;
```

```
5  jumlah_titik=hasil(3);
6  fileID = fopen(strcat(skenario,'SKENARIO_CH.csv'),
   'at');
7  fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4,
   t5,GAR,FAR,jumlah_titik);
8  fclose(fileID);
9
10 hasil = test_skenario2(18,20,14,0.38,4,'DATA/CENTER/');
11 GAR=hasil(1)/100*100;
12 FAR=hasil(2)/9900*100;
13 jumlah_titik=hasil(3);
14 fileID = fopen(strcat(skenario,'SKENARIO_CH.csv'),
   'at');
15 fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4,
   t5,GAR,FAR,jumlah_titik);
16 fclose(fileID);
17
18 hasil = test_skenario2(18,20,14,0.38,4,'DATA/BOTTOM/');
19 GAR=hasil(1)/100*100;
20 FAR=hasil(2)/9900*100;
21 jumlah_titik=hasil(3);
22 fileID = fopen(strcat(skenario,'SKENARIO_CH.csv'),
   'at');
23 fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4,
   t5,GAR,FAR,jumlah_titik);
24 fclose(fileID);
25
26 hasil =
   test_skenario2(18,20,14,0.38,4,'DATA/TOP_CENTER/');
27 GAR=hasil(1)/100*100;
28 FAR=hasil(2)/9900*100;
29 jumlah_titik=hasil(3);
30 fileID = fopen(strcat(skenario,'SKENARIO_CH.csv'),
   'at');
31 fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4,
   t5,GAR,FAR,jumlah_titik);
34 fclose(fileID);
35
36 hasil =
   test_skenario2(18,20,14,0.38,4,'DATA/TOP_BOTTOM/');
37 GAR=hasil(1)/100*100;
38 FAR=hasil(2)/9900*100;
39 jumlah_titik=hasil(3);
40 fileID = fopen(strcat(skenario,'SKENARIO_CH.csv'),
   'at');
```

```
41  fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4,
    t5,GAR,FAR,jumlah_titik);
42  fclose(fileID);
43
44  hasil =
    test_skenario2(18,20,14,0.38,4,'DATA/CENTER_BOTTOM/');
45  GAR=hasil(1)/100*100;
46  FAR=hasil(2)/9900*100;
47  jumlah_titik=hasil(3);
48  fileID = fopen(strcat(skenario,'SKENARIO_CH.csv'),
    'at');
49  fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4,
    t5,GAR,FAR,jumlah_titik);
50  fclose(fileID);
51
52  hasil =
    test_skenario2(18,20,14,0.38,4,'DATA/SELECTION/');
53  GAR=hasil(1)/100*100;
54  FAR=hasil(2)/9900*100;
55  jumlah_titik=hasil(3);
56  fileID = fopen(strcat(skenario,'SKENARIO_CH.csv'),
    'at');
57  fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4,
    t5,GAR,FAR,jumlah_titik);
58  fclose(fileID);
59
60  hasil = test_skenario2(18,20,14,0.38,4,'DATA/ALL/');
61  GAR=hasil(1)/100*100;
62  FAR=hasil(2)/9900*100;
63  jumlah_titik=hasil(3);
64  fileID = fopen(strcat(skenario,'SKENARIO_CH.csv'),
    'at');
65  fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4,
    t5,GAR,FAR,jumlah_titik);
66  fclose(fileID);
67  end
```

Kode Sumber 9.3 Uji Coba Pengaruh *Convex Hull*

```
1  function hasil =  UjiRunningTime()
2  hasil =
   test_skenario3(18,20,14,0.38,4,'DATA/SELECTION/');
3  GAR=hasil(1)/100*100;
4  FAR=hasil(2)/9900*100;
5  time=hasil(3);
```

```
6   fileID =
    fopen(strcat(skenario,'SKENARIO_RUNNINGTIME.csv'),
    'at');
7   fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4
    ,t5,GAR,FAR,time);
8   fclose(fileID);
9
10  hasil = test_skenario3(18,20,14,0.38,4,'DATA/ALL/');
11  GAR=hasil(1)/100*100;
12  FAR=hasil(2)/9900*100;
13  time=hasil(3);
14  fileID =
    fopen(strcat(skenario,'SKENARIO_RUNNINGTIME.csv'),
    'at');
15  fprintf(fileID,'%d,%d,%d,%f,%d,%f,%f,%f\n',t1,t2,t3,t4
    ,t5,GAR,FAR,time);
16  fclose(fileID);
17  end
```

Kode Sumber 9.4 Uji Coba terhadap Waktu *Running*

Lampiran 2. Perbandingan Metode yang Diusulkan

Tabel 9.1 Perbandingan Metode yang Diusulkan dengan yang Sudah Ada

No	Kategori	Metode yang Diusulkan	**Transforming minutiae for protecting fingerprint data**[7]	**Pair-polar coordinate-based cancelable fingerprint templates**[6]
1	Tujuan	Dibuat untuk mengamankan dan mencocokkan data sidik jari	Dibuat untuk mengamankan dan mencocokkan data sidik jari	Dibuat untuk mengamankan dan mencocokkan data sidik jari
2	Yang Diproses	Titik *minutiae*	Titik *minutiae*	Titik *minutiae*
3	Seleksi Titik	*Threshold Area* & *Convex Hull (Jarvis March)* area terluar, area tengah, area dalam	*Convex Hull (Graham Scan)* 3 x dari area terluar	*Threshold Area*
4	Berbasis Sektor	Ya (32 Sektor)	Ya (16 Sektor)	Ya (8 Sektor)

No	Kategori	Metode yang Diusulkan	Transforming minutiae for protecting fingerprint data[7]	Pair-polar coordinate-based cancelable fingerprint templates[6]
5	Transformasi	a. Rotasi	a. Rotasi	a. Rotasi
		b. Refleksi	b. Translasi	b. Dilatasi
		c. Translasi		
6	*Key*	Ada	Ada	Ada
7	Fitur Verifikasi	a. Jarak	a. Jarak	a. Jarak
		b. Sudut Orientasi	b. Sudut Orientasi	b. Sudut Orientasi
		c. Sudut antara garis dan sumbu x	c. Sudut antara garis dan sumbu x	c. Sudut antara garis dan sumbu x
		d. Sudut antara garis dan proyeksi garis arah orientasi terhadap sumbu x	d. sektor	d. sektor
		e. sektor		
8	Representasi data	(α, r, γ)	(d, β)	(r, α, β)

BIODATA PENULIS

Burhanudin Rasyid lahir di Blitar pada tanggal 09 Oktober 1994. Penulis menempuh pendidikan formal dimulai dari TK Al-Hidayah Bendowulung (2000-2001), MI Nurul Huda Bendowulung (2001-2007), SMPN Negeri 2 Blitar (2007-2010), SMAN 1 Blitar (2010-2013), dan S1 Teknik Informatika (2013-2017). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Penulis memiliki minat pada biometric terutama aplikasinya pada fingerprint. Penulis lebih suka berwirausaha dibidang IT khusunya *software house* maupun pembuatan product berteknologi. Penulis dapat dihubungi melaluli surel pribadi pada hanumuslem@gmail.com atau melalui surel formal pada burhanudin13@mhs.if.its.ac.id.