# ANALISIS, PERANCANGAN, DAN IMPLEMENTASI ADDITIONAL LAYER UNTUK GESTURE CLASH PADA MULTI TOUCH API

**Aisya Nafiisyanti[1], Kemas Rahmat Saleh W.[2], Bayu Munajat[2]**

[1,2,3]Fakultas Teknik Informatika Institut Teknologi Telkom, Bandung
[1]**aisya.nafiisyanti@yahoo.com**, [2]**rsw@ittelkom.ac.id**, [3]**bayumunajat@live.com**

**Abstrak**

**Pada requirement user, terdapat permintaan terhadap custom gesture yang tidak terdapat pada API sistem operasi bawaan perangkat. Untuk memenuhi kebutuhan permintaan user, dibuat custom gesture. Namun, muncul permasalahan baru ketika custom gesture yang digunakan mengalami clash. Untuk menghindari clash, user, dalam hal ini developer, dapat menggunakan aplikasi Additional Layer.**

**Aplikasi Additional Layer adalah aplikasi yang dibuat untuk mengetahui gesture-gesture yang mengalami clash sehingga user dapat menghindari penggunaan gesture yang saling clash secara bersamaan. Selain itu, aplikasi ini juga memprediksi kemungkinan clash dari suatu pola yang akan dijadikan gesture. Custom gesture yang dijasikan studi kasus adalah custom gesture keluaran SiGeR dengan nama SiGeR-v1.0.1.**

**Sistem aplikasi Additional Layer merupakan sistem yang kompleks dan menerapkan dua *design pattern*, yaitu Adapter Design Pattern dan Strategy Design Pattern. Selain itu, proses pengidentifikasian clash menggunakan algoritma Jaro Wrinkler Distance.**

**Sistem aplikasi ini dapat mengidentifikasi gesture yang mengalami clash dengan tingkat akurasi 95,45%. Selain itu, aplikasi ini juga membantu developer yang akan membuat gesture baru dengan cara melakukan pembandingan pola gesture baru tersebut dengan gesture yang sudah ada.**

**Kata kunci:** ***custom gesture, Adapter Design Pattern, Strategy Design Pattern, Jaro Wrinkler Distance*****, Additional Layer, SiGeR-v1.0.1**

**Abstract**

***User requirement on application sometimes contain custom gesture which is not available on device operating system API. To fill user requirement, custom gestures are made. But then, there are some problems when the custom gesture is clashed. To avoid the clash, user, developer in this case, could use Additional Layer application.***

***Additional Layer application is made to acknowledge themselves about some gestures that clashed, so user be able to avoid the using of clashed gesture at the same time.it also predicts clash possibility from a pattern which will become a gesture. Custom gesture that is researched is custom gesture launched by SiGeR, named SiGeR-v1.0.1.***

***Additional Layer application is a complex system and use two design patterns, Adapter Design Pattern and Strategy Design Pattern. Besides, identification process using Jaro Wrinkler Distance Algorithm.***

***This application be able to identify clashed gesture with 95.45% accuracy. Besides, this application also helps developer who will make new gesture by comparing the new gesture pattern with the old gesture.***

***Key words: custom gesture, Adapter Design Pattern, Strategy Design Pattern, Jaro Wrinkler Distance, Additional Layer, SiGeR-v1.0.1***

## 1. Pendahuluan

Pada teknologi perangkat berbasis sentuhan, terdapat API yang menjadi penghubung antara hardware dan software. API adalah Application Programming Interface yang menjadi dokumentasi, pendefinisi fitur *interface*, dan menjadi penghubung antara *software* dan *hardware* untuk aplikasi. Berdasarkan penjelasan Khandkar [3], dapat ditarik kesimpulan bahwa API adalah dokumentasi dari program yang memiliki kemampuan sebagai *gesture recognizer* dan *device driver*. Sedangkan menurut Akhena [1], API bertindak sebagai penyedia layanan yang mendukung dan melayani permintaan program komputer.

*Touch-based devices* menggunakan gesture untuk menerima perintah/ masukkan. Gesture secara etimologi adalah isyarat tubuh. Namun, pada bahasan ini, Gesture adalah adalah gerakan tangan dengan menyentuh permukaan dalam berbagai cara yang dilakukan oleh user untuk memberi perintah atau masukan pada alat. Fungsi gesture adalah untuk menerima interaksi yang intuitif[7].

Pada *software requirement* yang diberikan oleh *customer*, bisa saja terdapat fungsionalitas dengan gesture yang tidak tersedia pada Touch API, sehingga developer atau *programmer* melakukan

*override*[3,5] atau membuat gesture baru sendiri. Salah satu gesture yang dapat digunakan adalah gesture yang disediakan oleh SiGeR (Simple Gesture Recognition)[6]. SiGeR adalah penyedia gesture yang disarankan oleh salah satu penyedia API, yaitu Windows. Namun, muncul masalah baru, aplikasi tersebut mengalami gesture clash. Pada bahasan ini, gesture clash adalah bentrokan fitur antar-gesture yang mengakibatkan ambiguitas pada penggunaan gesture pada suatu fitur di dalam aplikasi. Contoh clash adalah ketika seorang user ingin menggunakan satu gesture untuk memberikan suatu perintah, *device* seolah-olah menerima perintah dari gesture tersebut dan gesture yang lain sehingga terjadi lebih dari satu aksi yang muncul. Hal ini terjadi karena pembacaan pola yang mirip/ hampir sama dari gesture yang saling clash. Akibatnya adalah munculnya kebingungan bagi user.

Untuk menghilangkan gesture clash, dilakukanlah *override* atau dengan kata lain, perbaikan dilakukan di akhir, tetapi hal ini tidak efektif karena membutuhkan biaya besar dan perlu ada perombakan aplikasi.

Berdasarkan permasalahan yang muncul di atas, tugas akhir ini dilakukan untuk membuat Additional Layer, yang merupakan aplikasi pendeteksi kemiripan pola gesture dengan menggunakan salah satu algoritma perbandingan dokumen, yaitu Jaro Wrinkler Distance, sehingga dapat mengantisipasi terjadinya gesture clash (perbaikan di awal) dan mempermudah pekerjaan developer atau *programmer* membangun aplikasi *touch-based*. Additional Layer ini akan dibuat dengan menerapkan beberapa *design pattern*, yaitu *Strategy Design Pattern* dan *Adapter Design Pattern*[5]. Kedua *design pattern* tersebut dipilih karena dapat mencakup kebutuhan developer untuk membuat kerangka algoritma, mengatasi pendeklarasian logika yang kompleks, tampilan yang dinamis, dan kebutuhan untuk mengubah algoritma yang beragam, tetapi berkaitan.

## 2. Gesture Clash

Gesture clash merupakan kesalahan pembacaan gesture dari sisi *library* (API) sehingga *library* baru yang dibuat tidak dapat mengenali suatu gesture atau mengenali gesture yang diterima oleh layar sentuh sebagai gesture yang lain. Hal ini terjadi karena *library* menemukan pendefinisian yang mirip dari beberapa gesture oleh suatu baris kode, sehingga *library* salah mengenali gesture. Akibat dari kesalahan pendefinisian gesture tersebut berakibat pada aksi yang dikeluarkan. Penggambaran kesalahan tersebut terdapat pada tabel berikut

Tabel 1: Pasangan Fungsionalitas dengan Gesture yang Benar

| Proses pembacaan gesture | Gesture | Fungsionalitas |
|---|---|---|
| Read input to G1 | G1 | F1 |
| Read input to G2 | G2 | F2 |
| Read input to G3 | G3 | F3 |

Gesture 1 dan gesture 2 memiliki kemiripan sehingga terjadi kesalahan.

Tabel 0: Pasangan Gesture dan Fungsionalitas yang Salah karena Gesture Clash

| Proses pembacaan gesture | Gesture | Fungsionalitas | Fungsionalitas yang seharusnya |
|---|---|---|---|
| Read input to G1 | G1 | F1 | F1 |
| Read input to G1 | G1 | F1 | F2 |

## 3. Jaro-Wrinkler Distance

Jaro Wrinkler Distance adalah algoritma varian dari Jaro Distance Metrik untuk mengukur kesamaan dua buah kata [4]. Nilai Jaro Wrinkler Distance adalah 0-1. Semakin tinggi nilai JWD, semakin mirip kedua kata tersebut, sedangkan semakin rendah nilai JWD, semakin berbeda kata yang dibandingkan. JWD cocok digunakan untuk membandingkan string singkat [4].

Algoritma JWD memiliki kompleksitas waktu *quadratic runtime complexity* yang efektif pada string pendek dan lebih efektif dibandingkan dengan algoritma *edit distance*. Algoritma ini memiliki 3 bagian utama pada prosesnya, yaitu

1. Menghitung panjang masing-masing kata yang dibandingkan
2. Menghitung jumlah karakter yang sama pada masing-masing kata
3. Menghitung nilai transposisi.

Ketiga bagian tersebut merupakan bagian tahapan untuk memperoleh nilai akhir JWD pada satu pasang kata. Tahapan untuk mendapatkan nilai akhir JWD terdiri dari dua proses, yaitu:

1. Menghitung jarak antara dua string, yaitu $|s_1|$ dan $|s_2|$
   Untuk menghitung jarak antara dua string tersebut (*dj*), terdapat rumus sebagai berikut

$$dj = \left(\frac{1}{3}\right) \text{x } \left(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m}\right) \quad (1)$$

Dengan,

*m* = jumlah karakter yang sama persis pada masing-masing string

$|s_1|$ = panjang karakter pada string pertama

$|s_2|$ = panjang karakter pada string kedua

*t* = jumlah transposisi

Biasanya, string yang dijadikan acuan adalah string pertama ($s_1$), sehingga perbedaan karakter akan dibandingkan dengan string pertama. Jumlah transposisi adalah jumlah posisi karakter yang saling bertukar berdasarkan pembandingan string kedua terhadap string pertama. Contoh, terdapat string 'DATE' dan 'TADE', kedua string tersebut memiliki karakter yang sama persis, yaitu 'D', 'A', 'T', 'E'. Namun, terdapat karakter yang posisinya tertukar, yaitu 'D' dan 'T'. Berdasarkan pembandingan string kedua terhadap string pertama, terdapat satu posisi karakter yang saling bertukar, sehingga nilai transposisi pada kedua string tersebut adalah 1. Pada contoh lain seperti string 'WASERI' dan 'WERI', jumlah karakter yang mirip adalah 4, yaitu 'W', 'E', 'R', 'I'. Meskipun terdapat karakter yang tidak sama, karakter 'W', 'E', 'R', 'I' pada string pertama memiliki urutan yang sama dengan string kedua, sehingga nilai transposisi untuk kedua string tersebut adalah 0.

2. Menghitung nilai Jaro Wrinkler Distance

Setelah mendapatkan nilai jarak antara dua string, tahap berikutnya adalah menghitung nilai Jaro Wrinkler Distance (*dw*) dengan rumus

$$dw = dj + (lp(1 - dj)) \quad (2)$$

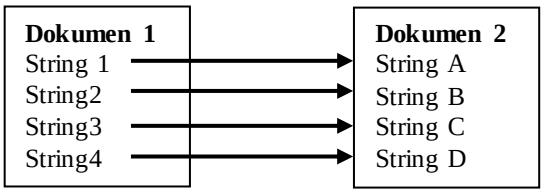
Dengan

*dj* = nilai jarak antara dua string

*l* = panjang prefiks umum (*prefix length*) di awal string maksimal 4 karakter (panjang jumlah karakter yang sama di awal string sebelum ditemukan ketidaksamaan maksimal 4 karakter)

*p* = *prefix scale* (konstanta *scaling factor*) dengan nilai 0.1

Bila pada dokumen-dokumen yang akan dibandingkan terdapat beberapa string (lebih dari 1), masing-masing string dibandingkan sesuai dengan urutannya.




Gambar 1: Pasangan String yang Dibandingkan

Untuk menghitung nilai JWD keseluruhan untuk pasangan dokumen tersebut, diambil jumlah nilai total dari masing-masing pasangan string dibagi jumlah string sesuai dengan rumus matematika (3).
Bila salah satu dokumen memiliki string yang jumlahnya lebih banyak, besar pembagi sejumlah banyaknya string pada dokumen yang memiliki string yang jumlahnya lebih banyak.

## 4. Adapter Design Pattern

Adapter design pattern adalah struktur pembantu yang dapat menghubungkan dua kelas atau lebih yang tidak saling sesuai, tetapi perlu untuk saling berhubungan[2]. Adapter dapat dianalogikan seperti MicroSD dengan *slot memory card* pada komputer. MicroSD tentu tidak dapat langsung dimasukkan pada *slot* karena ukurannya yang tidak sesuai. Maka dari itu, dibutuhkan *adapter card* yang dapat menyimpan MicroSD di dalamnya dan *adapter card* dapat masuk ke dalam *slot memory card*. Dengan begitu, microSD dan *slot memory card* tetap dapat berhubungan.

Adapter design pattern adalah struktur pembantu yang dapat menghubungkan dua kelas atau lebih yang tidak saling sesuai, tetapi perlu untuk saling berhubungan. Adapter dapat dianalogikan seperti MicroSD dengan *slot memory card* pada komputer. MicroSD tentu tidak dapat langsung dimasukkan pada *slot* karena ukurannya yang tidak sesuai. Maka dari itu, dibutuhkan *adapter card* yang dapat menyimpan MicroSD di dalamnya dan *adapter card* dapat masuk ke dalam *slot memory card*. Dengan begitu, microSD dan *slot memory card* tetap dapat berhubungan.

$$JWDtot = \frac{jd_{String1,stringA} + jd_{String2,stringB} + jd_{String3,stringC} + jd_{String4,stringD}}{4} \quad (3)$$

**5. Strategy Design Pattern**

Strategy design pattern berfungsi untuk mengelompokkan algoritma, mengenkapsulasi masing-masing kelompok tersebut, dan membuat kelompok-kelompok tersebut dapat saling bertukar[2]. Strategy juga memungkinkan algoritma bervariasi secara independent berdasarkan penggunanya.

Strategy design pattern dapat dianalogikan sebagai bagian pemroses pilihan tergantung dari kebutuhan kelas yang mengakses design pattern tersebut.

**6. Pengumpulan Data**

Data diambil dari aplikasi bernama SiGeR-v1.0.1. aplikasi ini merupakan aplikasi yang disarankan oleh sistem operasi Windows Phone pada developer yang ingin membuat atau memakai gesture baru di luar gesture yang sudah disediakan oleh API sistem operasi tersebut[6]. Aplikasi ini berbentuk simulator untuk menguji coba gesture yang sudah dibuat. User dari aplikasi ini juga dapat menambahkan gesture baru pada aplikasi dengan cara manambahkan baris kode dan beberapa file dengan ekstensi .ink sebagai library bentuk gesture pada beberapa class pembangun aplikasi.

Data yang diambil dari aplikasi ini berupa vektor yang ditangkap oleh aplikasi ketika user memasukkan gesture dan masukkan tersebut terdeteksi sebagai suatu gesture. Contoh bentuk vektor adalah seperti berikut ini

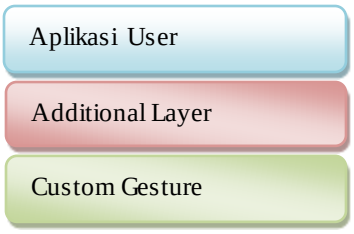
RD,RD,RD,RD,D,RD,D,RD,RU,RU,RU,RU,U,RU,
RU,U,RU,RU,RU,RU,RU,

Terdapat 12 gesture yang akan diuji, pada masing-masing gesture diambil 2 set data, yaitu 10 vektor untuk masing-masing gesture dan 100 vektor untuk masing-masing gesture, sehingga total terdapat **120 vektor dan 1200 vektor** yang dijadikan data mentah. Pada proses pengambilan data, terdapat masukkan yang tidak teridentifikasi sebagai sebuah gesture. Vektor dari masukkan tersebut tidak diambil sebagai data (diabaikan).

Untuk pengujian hasil perhitungan nilai Jaro Wrinkler Distance, nilai keputusan ya (terjadi clash) atau tidak (tidak terjadi clash) dibandingkan dengan hasil percobaan secara manual terhadap aplikasi SiGeR-v1.0.1. pada masing-masing gesture, diambil dua data set hasil percobaan masukkan yang terdeteksi sebagai gesture yang dimaksud. Dari percobaan tersebut diperoleh persentase terjadi clash atau tidak. Angka persentase tersebut yang akan dibandingkan dengan hasil yang diperoleh dari perhitungan Jaro Wrinkler Distance.

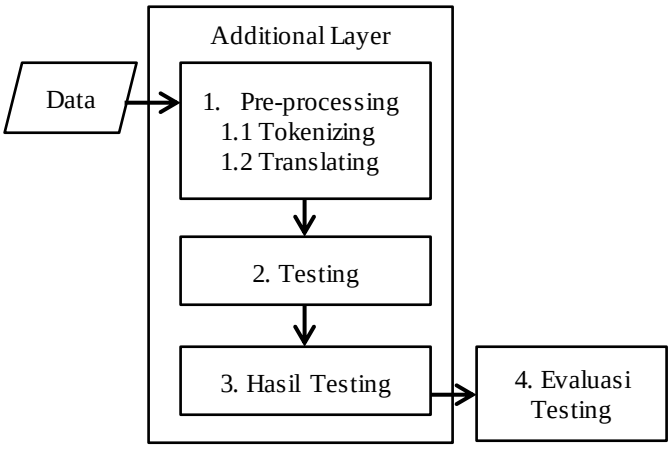
**7. Perancangan Sistem**

Additional Layer adalah nama dari aplikasi yang dibuat untuk mengidentifikasi gesture clash pada custom gesture. Aplikasi ini memudahkan developer memeriksa gesture baru (custom gesture) yang dibuat untuk menghindari gesture clash. Aplikasi ini berfungsi sebagai pengontrol yang akan mengeluarkan hasil perhitungan dari perbandingan beberapa gesture yang dipilih oleh user dan mengeluarkan pesan saran bila terdapat indikasi gesture clash. Additional Layer ini menggunakan design pattern sebagai pola kelas yang ada di dalam layer tersebut. Nantinya, developer yang akan membuat aplikasi dengan gesture buatan sendiri akan menggunakan aplikasi ini sebagai acuan saran untuk memilih gesture yang tidak akan saling clash.Additional Layer bersifat sebagai aplikasi perantara antara custom gesture dengan aplikasi yang akan menggunakan custom gesture tersebut, sehingga posisi Additional Layer seperti berikut




Gambar 2:Posisi Additional Layer terhadap Custom Gesture

Secara umum, sistem pada aplikasi Additional Layer tergambar pada diagram sistem berikut




Gambar 3: Rancangan Keseluruhan Sistem Aplikasi Additional Layer

**8. Skenario Pengujian**

Terdapat dua fungsionalitas yang dimiliki oleh system, yaitu GetSolution dan CheckSimilarity. Masing-masing fungsionalitas akan diuji dengan skenario pengujian sebagai berikut

Pengujian untuk fungsionalitas GetSolution

1. Pengambilan data berupa hasil percobaan sebanyak 20 kali dan 100 kali terhadap masing-masing gesture pada aplikasi SiGeR-v1.0.1 untuk mendapatkan

kesimpulan suatu gesture mengalami clash dengan gesture lain atau tidak. Hasil ini akan digunakan sebagai acuan pembanding keberhasilan sistem yang dibuat.

2. Pengambilan data berupa pembandingan seluruh gesture. Terdapat 66 pasang gesture yang akan diuji dan hasilnya berupa pernyataan clash atu tidak clash.
3. Pembandingan hasil yang diperolah dari aplikasi SiGeR-v1.0.1 dengan hasil dari aplikasi Additional Layer.

Pengujian untuk fungsionalitas CheckSimilarity

1. Membuat dua pola gesture baru. Dua pola baru tersebut adalah pola gesture 'Zoro' dan 'Love'. Pola gesture 'Zoro' dirancang akan mengalami clash dengan salah satu gesture yang sudah ada, yaitu Scratch Out. Untuk mengeset pola supaya terjadi clash dengan gesture Scratch Out, pola gesture 'Zoro' dibuat hampir sama seperti pola gesture Scratch Out. Pola gesture Zoro adalah

   **Rights+LeftDowns+Rights**

2. Membuat pola gesture kedua, yaitu gesture 'Love'. Pola gesture ini dirancang tidak memiliki kesamaan dengan pola gesture lain yang sudah ada. Pola gesture Love adalah

   **LeftUps+Lefts+RightDowns+CornerTick +RightUps+LeftsDowns**
3. Menjadikan kedua pola gesture pada dua poin di atas sebagai masukkan pada aplikasi dan membandingkan hasil perhitungan yang dikeluarkan dengan skenario, apakah sesuai atau berbeda?

## 9. Hasil Pengujian

Dua tabel di bawah ini adalah hasil pengujian terhadap fungsionalitas GetSolution

Tabel 3: Hasil Pengujian Terhadap Aplikasi SiGeR

| No. | Pasangan Gesture yang Mengalami Clash | |
|---|---|---|
| | **Hasil Data Set 1** | **Hasil Data Set 2** |
| 1. | Right Bracket – Question | Right Bracket – Question |
| 2. | Rectangle-Square | Rectangle – Square |
| 3. | Scratch Out-Curlicue | Scratch Out – Curlicue |
| 4. | Star – Double Circle | Star – Double Circle |

Tabel 4: Hasil Pengujian Terhadap Additional Layer

| No. | Pasangan Gesture yang Mengalami Clash |
|---|---|
| 1. | Rectangle-Square |

Hasil ini merupakan hasil pengujian terhadap dua data set yang digunakan

Untuk hasil pengujian terhadap fungsionalitas CheckSimilarity, disajikan pada tabel di bawah ini

Tabel 5: Hasil Perhitungan Pasangan Pola Gesture Love dengan Gesture Lain

| No. | Gesture yang dibandingkan | Hasil Perbandingan | |
|---|---|---|---|
| | | Data Set 1 | Data Set 2 |
| 1. | Rectangle | 0.14930 | 0.17411 |
| 2. | Square | 0.14216 | 0.16530 |
| 3. | Triangle | 0.15066 | 0.15717 |
| 4. | Right Bracket | NaN | NaN |
| 5. | Scratch Out | NaN | NaN |
| 6. | Curlicue | 0.08508 | 0.11407 |
| 7. | Double Curlicue | 0.05162 | 0.06482 |
| 8. | Circle | 0.11835 | 0.07272 |
| 9. | Double Circle | 0.06080 | 0.04664 |
| 10. | Question | 0.20853 | 0.13907 |
| 11. | Check Mark | 0.15012 | NaN |
| 12. | Star | 0.09101 | 0.09766 |

Keterangan: nilai NaN disebabkan oleh angka yang terlalu kecil.

## 10. Analisis Pengujian

Pada pengujian fungsionalitas GetSolution, Berdasarkan hasil, terdapat gesture yang berhasil diidentifikasi mengalami clash, tetapi ada juga yang tidak teridentifikasi. Pasangan clash yang tidak teridentifikasi adalah Scratch Out-Curlicue dan Right Bracket-Question.

Pasangan gesture ini tidak teridentifikasi pada aplikasi Additional Layer karena pola umum kedua gesture tersebut berdasarkan **urutan dan panjang string berbeda**. Hal tersebut terlihat pada pola berikut

Scratch Out : R,L,R,L
Curlicue : RU, U, L,LD, D, R, RU,

Kondisi ini membuat hasil perhitungan JWD tidak mencapai batas threshold dan cenderung sangat kecil. Seperti yang sudah dijelaskan pada bab Landasan Teori, algoritma Jaro Wrinkler membandingkan dua string pada urutan yang sama, sehingga pada data Scratch Out dan Curlicue, pasangan string yang menghasilkan nilai lebih dari 0 adalah R-RU dan L-LD. Sedangkan pasangan string lain menghasilkan nilai = 0. Namun, hal ini tidak cukup membantu menaikkan nilai hasil akhir perhitungan karena hasil total dibagi dengan jumlah string terpanjang dari salah satu vektor. Hal yang sama juga terjadi pada pasangan lain.

Pada aplikasi SiGeR-v1.0.1, kedua gesture tersebut dianggap mengalami clash karena pola vektor secara keseluruhan cenderung mirip dengan beberapa vektor tambahan pada Scratch Out. Aplikasi tersebut dapat mengabaikan tambahan vektor tersebut. Hal tersebut tidak dapat dilakukan oleh algoritma Jaro Wrinkler Distance. Selain itu, vektor bukan satu-satunya hal yang mempengaruhi pembacaan koordinat. Terdapat beberapa variabel yang nilainya bergantung pada masukkan koordinat dan konstanta yang sudah ditetapkan pada kode aplikasi.
Persentase keberhasilan perhitungan Jaro Wrinkler Distance adalah sebagai berikut

$$Persentase\ keberhasilan = \frac{63}{66} \text{ x } 100\%$$
$$Persentase\ keberhasilan = \mathbf{95,45}\%$$

Pada skenario gesture 'Zoro', hasil yang didapatkan dari percobaan gesture Zoro, gesture yang diindikasikan akan mengalami clash tidak muncul seperti pada skenario. Nilai perhitungan tersebut tidak mencapai nilai threshold. Pola gesture Zoro yang masih berbentuk potongan kode ditranslasikan ke bentuk vektor dan hasilnya seperti berikut

Zoro : R, LD, R
Scratch Out : R, L, R, L

Bila dilihat dari urutan vektor kedua gesture di atas, seharusnya nilai hasil perbandingan mencapai 0.9, tetapi hal itu tidak sesuai dengan hasil perhitungan pada aplikasi Additional Layer. Hal ini karena vektor gesture Zoro dibandingkan dengan seluruh data vektor gesture Scratch Out, dan gesture lain, untuk mendapatkan nilai perbandingan paling global. Hal ini mengakibatkan nilai tertinggi dari seluruh hasil percobaan tidak mencapai nilai threshold. Bila nilai threshold diturunkan/diperkecil, akan ada gesture yang dianggap mengalami clash dengan gesture Zoro. Namun, hal ini menunjukkan penurunan tingkat akurasi.

Berdasarkan hasil yang telah diperoleh tersebut dapat dinyatakan bahwa tingkat keberhasilan fungsionalitas CheckSimilarity pada gesture Zoro adalah **0%**.

Sedangkan pada skenario gesture 'Love', hasil yang didapatkan dari percobaan gesture Love, tidak terdapat gesture yang diindikasikan akan mengalami clash dengan gesture Love dan hal ini sesuai dengan rancangan skenario. Tidak ada nilai perhitungan yang lebih besar dari nilai threshold karena tidak terdapat kemiripan pola vektor. Berdasarkan hasil yang telah diperoleh tersebut dapat dinyatakan bahwa tingkat keberhasilan fungsionalitas CheckSimilarity pada gesture Love adalah **100%**.

## 11. Kesimpulan

Terdapat kesimpulan yang dapat diambil dari tugas akhir ini, yaitu

1. Penyederhanaan data vektor dengan cara menghilangkan vektor yang berulang dalam urutan dapat meningkatkan nilai akurasi.
2. Algoritma Jaro Wrinkler Distance bekerja berdasarkan urutan pasangan. Hal ini membuat string vektor yang 'terselip' dapat memengaruhi nilai perhitungan.
3. Pada algoritma Jaro Wrinkler Distance, tepatnya pada perhitungan nilai jarak antara dua string, perlu ditambahkan bilangan yang nilainya sangat kecil untuk menghindari pembagian dengan 0.
4. Hasil pengujian pada fungsionalitas GetSolution menunjukkan akurasi sebesar 95,45% (dengan pasangan toleransi). Akurasi tidak mencapai 100% karena terdapat pasangan yang tidak terdeteksi sebagai clash. Hal ini terjadi karena panjang vektor dari kedua gesture berbeda jauh (panjang pola satu vektor setengah dari panjang vektor lain) dan urutan vektor yang berbeda (seperti yang telah disebutkan pada poin 3) sehingga hasil perhitungan tidak mencapai nilai threshold.
5. Hasil pengujian pada fungsionalitas CheckSimilarity tidak semuanya sesuai dengan rencana di skenario pengujian. Pada gesture Zoro, pola gesture yang diajukan tidak diidentifikasi mengalami clash dengan gesture Scratch Out. Hal ini terjadi karena nilai JWD pada kedua pola tidak mencapai nilai threshold. Pada gesture Love, pola yang diajukan tidak memiliki kemiripan dengan gesture manapun sehingga hasil perhitungan JWD tidak mencapai nilai threshold. Berdasarkan kondisi tersebut, gesture Love dinyatakan tidak mengalami clash dengan gesture manapun.
6. Penentuan nilai threshold yang tinggi (90%) untuk hasil perhitungan Jaro Wrinkler Distance untuk mencapai nilai batas yang akurat tidak menghasilkan pengelompokkan data yang optimal karena terdapat data yang tidak terkelompokkan pada kelas yang tepat.

**Daftar Pustaka:**


[1] Akhena, Mirza. 2009. *Perancangan, Implementasi, dan Analisis 3D Graphics Framework Extensi M3G Framework*. Bandung, Jurusan Teknik Informatika, Institut Teknologi Telkom.

[2] Gamma, Erich., Richard Helm, Ralph Johnson, Jhon Vlissides. 1993. *Design Pattern: Abstraction and Reuse of Object-Oriented Design*. Springer-Verlag, Heidelberg.

[3] Khandkar, Shahedul Huq and Frank Maurer. 2011. *A Domain Specific Language to Define Gestures for Multi-Touch Applications*. Canada, Departement of Computer Science, University of Calgary.

[4] Kurniaty, Anna. Puspudjati, Sulistyo. Rahman, Sazali. 2012. *Implementasi Algoritma Jaro Wrinkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia*. Indonesia, Universitas Gunadharma.

[5] Larman Craig. 2004. *Applying UML and Patterns-An introduction to Object-Oriented Analysis and Design and Iterative Development*. United States, Prentice Hall.

[6] Swigart, Scot. 2005. *Easily Write Custom Recognizer for Your Tablet PC Applications*. United States, MSDN Magazine. http://msdn.microsoft.com/en-us/library/aa480673.aspx

[7] Thornlund, Michael. 2007. *Gesture Analyzing for Multi-Touch Screen Interfaces*. Lulea University of Technology. Department of Skelleftea Campus. Division of Leisure and Entertainment