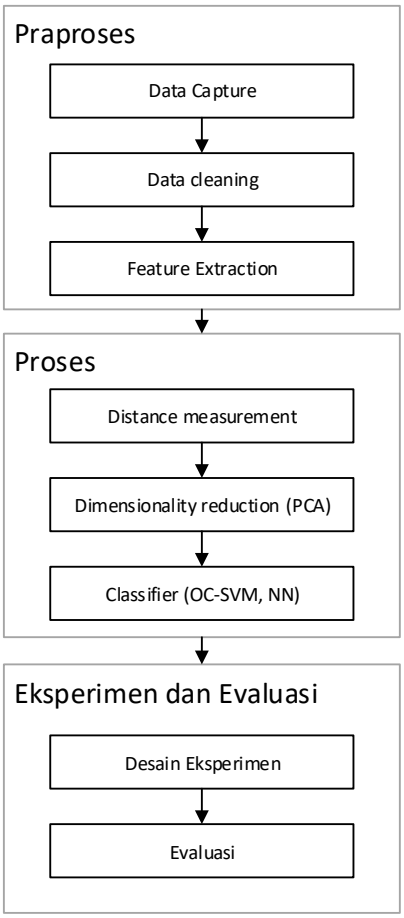
BAB III
METODOLOGI PENELITIAN

3.1. Model Penelitian

Penelitian ini dilakukan dalam beberapa tahapan yang dimodelkan dalam Gambar 3.1.



Gambar 3.1. Alur metodologi penelitian (Shen et al., 2013).

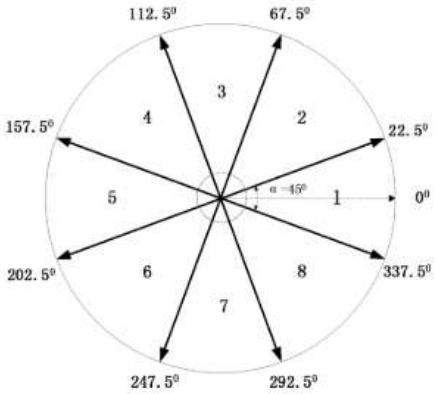
Pada Gambar 3.1 dipaparkan alur metodologi yang digunakan oleh penulis. Metodologi yang dipakai terdiri dari 8 tahap. Pada praproses dilakukan 3 tahap yaitu pengumpulan data (Shen et al., 2013) dengan merekam pergerakan tetikus, *data cleaning* untuk mengurangi galat data (Dasu & Johnson, 2003), dan ekstraksi fitur untuk mendapatkan informasi *behavioral* biometrik tetikus (Shen et al., 2013).

Pada proses utama dilakukan 3 tahap yaitu *distance measurment* (Chang, Desoky, Ouyang, & Rouchka, 2009; Sakoe & Chiba, 1978), *dimensionality reduction* (Jolliffe, 2002), dan *classifier* (Schölkopf et al., 2000). *Distance measurement* yang digunakan penulis adalah *manhattan distance* dan *dynamic time warping distance*. *Dimensionality reduction* menggunakan *principal component analysis* (PCA). *Classifier* yang digunakan adalah *oneclass support vector machine* (OC-SVM) sebagai *classifier* utama dan *neural network* sebagai pembanding.

Setelah klasifikasi selesai, penulis melakukan eksperimen yang dirancang sedemikian rupa, untuk mengetahui hasil dari praproses dan proses. Setelah didapatkan hasil maka dilakukan evaluasi menggunakan metode *error-rate* (Shen et al., 2013), yaitu *false acceptance rate* dan *false rejection rate*. Dilakukan pula evaluasi kecepatan autentikasi dalam satuan milisekon. Masing-masing tahapan dijelaskan pada subbab berikutnya.

3.2. *Data Capture*

Perekaman data dilakukan dengan merekam operasi tetikus yang telah didesain. Operasi pergerakan tetikus didesain menggunakan C# dengan model 8 arah pergerakan tetikus seperti Gambar 3.2. Jarak tempuh untuk tiap task dalam aplikasi ini di bagi menjadi 3 kategori jarak yaitu dekat, sedang, dan jauh



Gambar 3.2. 8 arah pergerakan tetikus. Sector 1 meliputi semua *task* di antara -22.5 derajat dan 22.5 derajat.

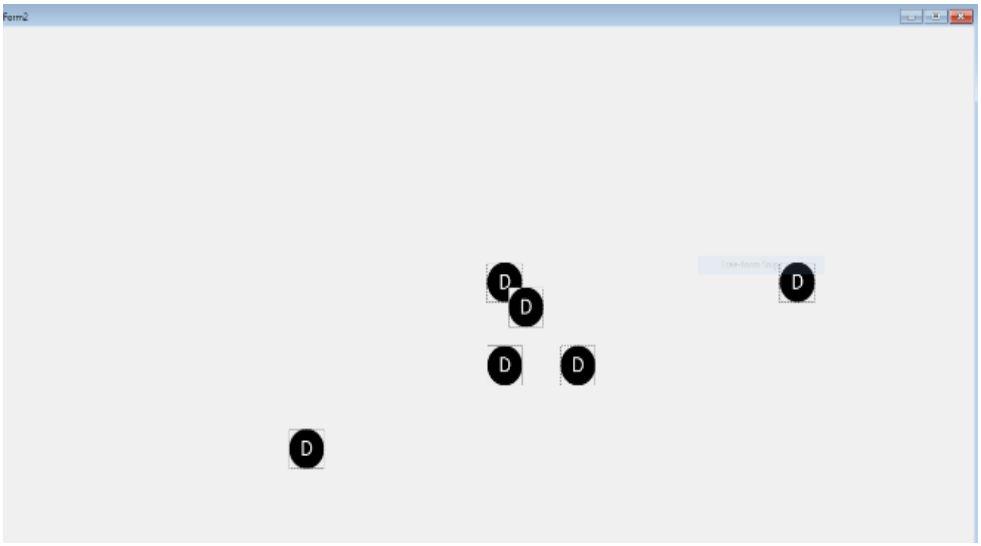
Task yang diberikan yaitu 8 *single click*, 8 *double click*, dan 16 *mouse movement* yang menyertainya. Total 32 *task* yang direkam pada penelitian ini. Seperti terlihat pada gambar 4.1, Penulis membuat aplikasi C# untuk merekam pergerakan tetikus berdasarkan 8 arah. Setiap arah diwakili oleh 2 gerakan, satu untuk *singleclick* dan lainnya *doubleclick*.

Task yang dibuat oleh penulis juga didasarkan pada jarak dan arah pergerakan. Jumlah 8 arah gerakan yang dibagi menjadi 3 jangkauan; dekat, sedang, dan jauh.

Tabel 3.1. Arah dan jarak gerakan tetikus dalam model *task* perekaman

Gerakan No.	Arah	Jarak (Pixel)
1	↑	100
2	→	400
3	↙	700
4	↗	700
5	↓	400
6	↖	524
7	↘	100
8	←	100

Tabel 3.3 berisi desain *task* yang akan di aplikasi dalam percobaan. Terdiri dari 8 jenis arah dan 3 jenis jarak. Jarak didasarkan pada model Shen et al. dengan kombinasi 3 gerakan dekat berjarak 100 pixel, 3 gerakan sedang berjarak 400 dan 524 pixel, dan 2 gerakan jauh berjarak 700 pixel.



Gambar 3.3. Desain task yang dibuat oleh penulis berdasarkan Gambar 4.1 dan Table 4.1

Pada Gambar 3.3, desain yang dibuat oleh penulis berupa aplikasi layer penuh dengan resolusi 1280x720 pixel. Setiap lingkaran yang mucul merupakan *task*. Setiap lingkaran merepresentasikan 1 *task* klik, dimana lingkaran hitam tanpa huruf adalah *task singleclick* dan lingkaran hitam dengan huruf D adalah *task doubleclick.*

Pada penelitin ini Data didapatkan dari hasil perekaman menggunkan software yang dibuat oleh penulis dengang menggunakan desain menurut Shen et al. Data masukan berupa teks berbasis csv dengan delimiter spasi. Setiap subjek akan memiliki 100 buah berkas yang masing-masing merepresntasikan 1 kali perekaman data. Jumlah data dalam satu berkas tidak ditentukan akan tetapi jumlah *task* dalam percobaan adalah tetap.

Data hasil perekaman terdiri atas 4 fitur, sebagai berikut:

1. *Mouse event*
   Menunjukkan jenis *event* yang terekam
2. *X coordinate*
   Koordinat sumbu x dalam satuan pixel
3. *Y coordinate*
   Koordinat sumbu y dalam satuan pixel

4. *Timestamp*

Penghitung waktu dalam satuan milisekon

Event, x, y, timestamp
512,166,95,1388609
512,169,95,1388625
512,170,95,1388625
512,171,95,1388640
512,172,95,1388640
512,172,95,1388640
513,172,95,1388734
514,172,95,1388828

Gambar 3.4. Data contoh data hasil perekaman

Seperti terlihat pada Gambar 3.4, data hasil rekaman simpan dalam format teks dengan delimiter koma, sesuai dengan format csv. Data terbagi menjadi empat kolom yang secara berurutan merupakan kode aksi tetikus, koordinat x, y, dan *timestamp* dalam bentuk milisekon. Satu task dalam perekaman akan menghasilkan beberapa kode aksi 512 yang akan diakhiri dengan kode 513 dan 514 secara berurutan. kode 512 dan 513 merupakan satu paket aksi yakni *singleclick* kiri. Pada sebuah aksi *aksi singleclick* pasti ada 2 *event_msg* yang diterima yaitu *down* dan *up*. Kode aksi tetikus beserta penjelasannya pada Tabel 3.2.

Tabel 3.2. Kode aksi tetikus

Kode	**Keterangan**
512	**Mouse movement event.**
513	**Left click Down event**
514	**Left click Up event**

Kode yang digunakan berdasarkan *mouse_event* yang diterima dari *event_MSG* Microsoft Windows.

Setiap partisipan diharuskan merekam sebanyak 150 kali dalam interval 10 sampai 20 rekaman per hari. Setiap file rekaman mewakili satu vektor dalam data fitur hasil ekstraksi. Dengan demikian jumlah file per subjek berjumlah 150 buah yang mewakili 150 vektor fitur *behaviour* biometrik tetikus.

3.3. *Data Cleaning*

Dalam data yang telah didapatkan tentunya terdapat hal-hal yang dapat membuat proses eksperimen tidak berjalan baik. Untuk memastikan data hasil rekaman yang didapatkan memenuhi kriteria , tidak redundan, dan bebas galat maka perlu dilakukan *data cleaning*. Proses *data cleaning* yang dilakukan terpapar pada Gambar 3.5.

Gambar 3.5. Bagan *data cleaning*

Data cleaning pertama pengecekan redundansi data atau adanya data berganda pada setiap berkas. Untuk setiap baris data yang redundan akan dihapus hingga hanya satu data yang tersisa. Dapat dilihat pada Gambar 3.6 , baris yang di tebalkan dan digarisbawahi merupakan data redundan yang perlu dihapus salah satunya.

Event, x, y, timestamp
512,166,95,1388609
512,169,95,1388625
512,170,95,1388625
512,171,95,1388640
512,172,95,1388640
512,172,95,1388640
513,172,95,1388734
514,172,95,1388828

Gambar 3.6. Contoh redundansi data

Data cleaning kedua, pengecekan galat *timestamp* dimana input memiliki waktu yang sama, sehingga hasil penghitungan waktu menghasilkan 0 sekon. Ini berarti penghitungan fitur laju nantinya, akan menghasilkan *infinite error*. Maka dari itu, penulis menghapus entri yang berdekatan dengan selisih *timestamp* 0 milisekon. Akan disisakan 1 entri data dengan selisih terjauh. Pada Gambar 3.7 data yang di garisbawahi memiliki timestamp yang sama sehingga salah satu harus dihapus.

Event, x, y, timestamp
512,166,95,1388609
512,169,95,1388625
512,170,95,1388625
512,171,95,1388640
512,172,95,1388640
513,172,95,1388734
514,172,95,1388828

Gambar 3.7. Contoh data *timestamp* kembar.

Data cleaning ketiga, pembersihan data movement yang tidak sengaja terekam saat proses *event* klik. Setiap *event* klik terdiri dari *event mouse-down* dan *mouse-up*. Diantara kedua *event* tersebut tidak boleh ada rekaman event lain. Penyisipan data Ini bisa terjadi karena ada pergerakan micro yang terekam oleh sistem tetapi dihiraukan oleh fungsi *event mouse* di Windows. Hal ini bisa menyebabkan data dianggap sebagai *drag-and-drop* daripada *click*. Maka dari itu, *event movement* di antara *mouse-down* dan *mouse-up* dihilangkan.

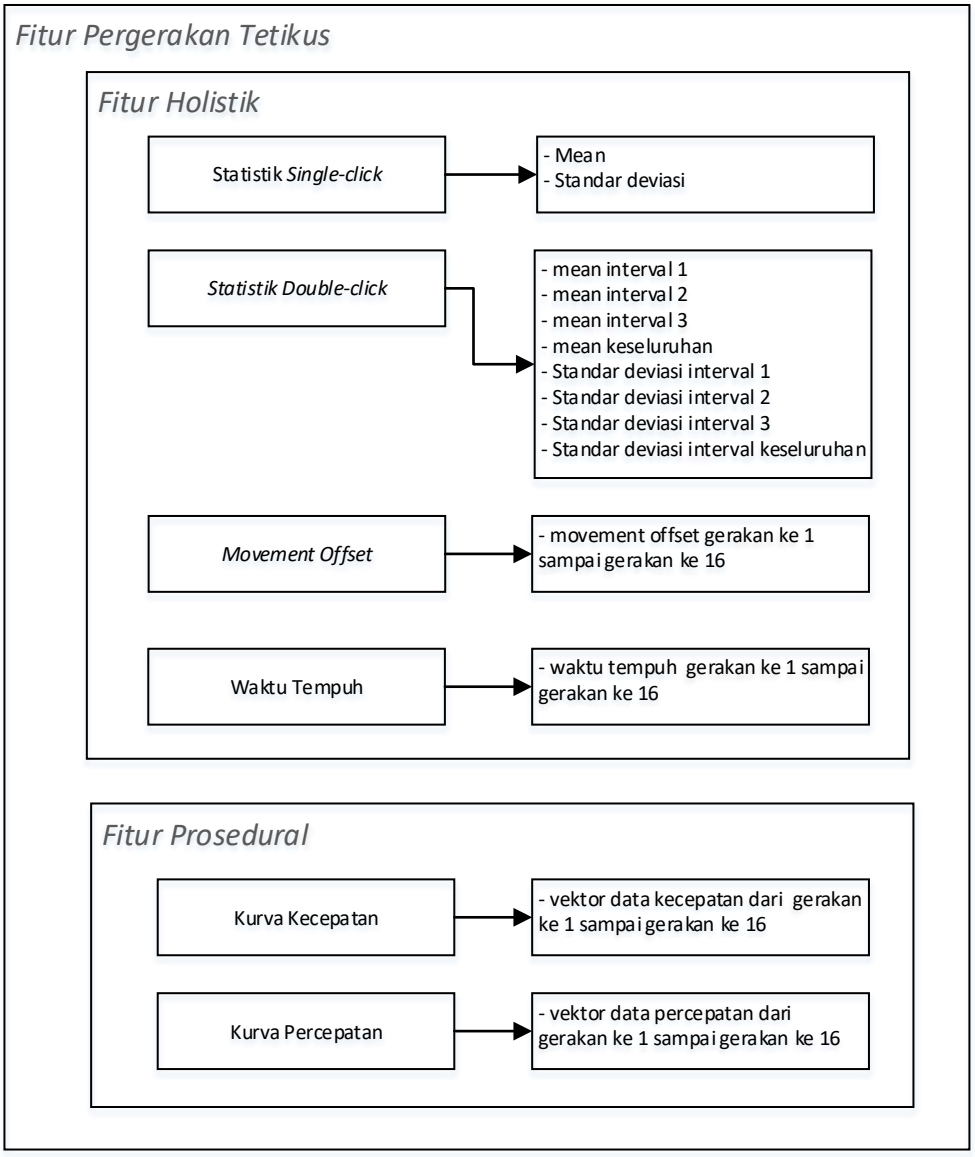
Event, x, y, timestamp
512,166,95,1388609
512,169,95,1388625
512,171,95,1388640
513,172,95,1388734
512,172,95,1388745
514,172,95,1388828

Gambar 3.8. Contoh event yang tidak seharusnya

Pada Gambar 3.8 ada *event* 512 pada baris yang ditebalkan yang berarti adanya pergerakan, namun *event* ini berada di antara *event* 513 dan 514 yang merupakan *downclick* dan *upclick*. Hal ini tidak diinginkan karena tidak boleh ada *event* di antara *downclick* dan *upclick* pada *task single-click*.

3.4. *Feature Extraction*

Data hasil rekaman yang telah melalui praproses belum bisa digunakan untuk masukan *classifier*, diperlukan ekstraksi fitur sehingga data menjadi berarti. Fitur yang dimaksud merupakan fitur pergerakan mouse yang telah didesain sebelumnya. Penulis menggunakan fitur yang dirancang oleh Shen et al. pada penelitian ini.



Gambar 3.9. Bagan fitur Pergereakan Tetikus.

Data akan diekstrak dan dibagi kedalam 2 kategori yaitu fitur holistik dan fitur prosedural. fitur holistik terdiri atas 42 fitur berkaitan dengan statistik jarak dan waktu. Fitur prosedural terdiri atas 32 fitur berkaitan dengan kelajuan dan percepatan sehingga total adalah 74 fitur. Proses ekstraksi di ilustrasikan pada Gambar 3.9.

Dari 7 subjek didapatkan 1050 data pergerakan tetikus. Semua data ini dipakai dalam penelitian. Data tersebut akan digunakan dalam ekstraksi fitur yang sudah didesain sebelumnya. Detail fitur pada Tabel 3.3 berdasarkan Shen et al.

Tabel 3.3. Fitur pergerakan tetikus (Shen et al., 2013).

Kategori	Fitur tetikus	Keterangan	Satuan	Jumlah fitur
Fitur Holistik	Statistik *single-click*	Mean dan standar deviasi dari keseluruhan waktu operasi *single-click*	Milisekon(ms)	2
	Statistik *double-click*	Mean dan standar deviasi dari satu keseluruhan waktu dan tiga interval operasi *double-click*	Milisekon(ms)	8
	Movement offset	Jarak antara lintasan yang ditempuh secara praktik dan lintasan ideal (gerak lurus) untuk setiap gerakan	Pixel	16
	Waktu tempuh	Waktu antara titik mula dan titik akhir di setiap gerakan	Milisekon(ms)	16
Fitur Prosedural	Kurva kecepatan	Kecepatan pergerakan yaitu rasio antara jarak antar titik yang bersebelahan, dibagi dengan interval waktu diantara dua titik tersebut, untuk setiap pergerakan.	Pixel/ms	16
	Kurva percepatan	Percepatan pergerakan yaitu rasio antara kecepatan antar titik yang bersebelahan, dibagi dengan interval waktu diantara dua titik tersebut, untuk setiap pergerakan.	$Pixel/ms^2$	16

Berdasarkan pemaparan sebelumnya, fitur-fitur yang akan diekstrak dibagi menjadi dua kategori yaitu :

1. Fitur Holistik: fitur yang menggolongkan keseluruhan sifat dari *mouse behaviour* saat interaksi terjadi, seperti statistik *single-click* dan *double-click*.
2. Fitur Prosedural: fitur yang menggambarkan secara detail proses dinamis dari *mouse behaviour*, seperti kurva kecepatan dan percepatan.

Fitur ini dipilih berdasarkan penelitian sebelumnya (Shen et al., 2013) dengan kesimpulan bahwa fitur-fitur ini merupakan fitur yang signifikan dalam pembuatan autentikasi menggunakan pergerakan tetikus. Fitur yang diekstrak seperti dijelaskan pada Tabel 3.3 dan di simpan dalam bentuk vector untuk setiap *task* pergerakan tetikus yang dijalankan

1. Fitur Holistik
   a) Statistik *Single-click*

      Statistik *single-click* dicari dengan cara menghitung waktu yang terlewati antara *mouse-down* dan *mouse-up event*.

Downclick ← **513**,707,313,705576921
Upclick ← **514**,707,313,705577046

Gambar 3.10. *Mouse event* dan *timestamp* yang berkaitan.

Seperti terlihat pada Gambar 3.10, *mouse event* yang terjadi adalah *single-click*, dengan kode 513 sebagai *downclick* dan 514 sebagai *upclick*. Untuk bisa mendapatkan nilai staistik mean dan standar deviasi penulis harus mengumpulkan nilai waktu interval antara 513 dan 514 untuk setiap *single-click* yang ada yaitu 8 buah. Pertama, penulis menghitung rentang waktu yang terjadi dengan mengurangi *timestamp* 514 dan *timestamp* 513.

$$interval\ = x_{i+1}[4] - x_i[4]$$

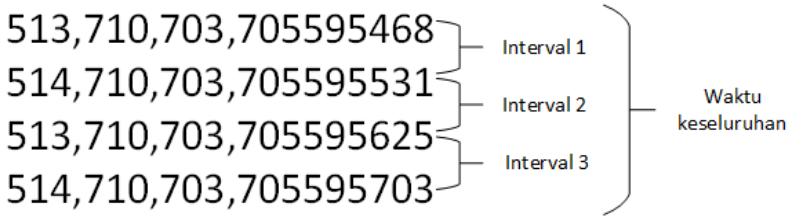
Data yang diambil yaitu data kolom ke empat yang merupakan kolom timestamp. Data waktu *single-click* yang terkumpul disimpan dalam vektor, kemudian dihitung mean dan standar deviasinya.

```
> temp
[1] 78 78 78 93 78 94 94 63
```

Gambar 3.11. Vektor *temp* berisi nilai waktu *single-click* dari 8 *task* yang telah dibuat

b) statistik *Double-click*

Statistik *double-click* dicari dengan cara menghitung waktu yang dibutuhkan dalam melakukan aksi *double-click. Double click* terdiri dari 4 aksi beruntun , yang didalamnya terdapat 3 interval waktu.



Gambar 3.12. Empat aksi dan 3 interval dalam sebuah *double click*

Terlihat pada Gambar 3.12, dari satu kali aksi *double click* didapatkan vektor waktu interval 1, interval 2, dan interval 3. Dari kumpulan data ini didapatkan 8 buah data yaitu mean masing-masing interval dan keseluruhan interval, serta standar deviasi masing-masing interval dan standar deviasi keseluruhan interval.

c) *Movement Offset*

Movement offset didapatkan dengan menghitung selisih jarak total pergerakan tetikus yang ditempuh oleh user dan jarak optimum yang mungkin. Jarak optimum dapat dicari dengan rumus *euclidian distance*:

$$d_{opt} = \sqrt{(|x_{awal} - x_{akhir}|)^2 + (|y_{awal} - y_{akhir}|)^2} \qquad (1)$$

Sedangkan untuk mencari jarak pergerakan tetikus oleh pengguna sebenarnya, penulis menggunakan rumus *euclidian distance* setiap dua titik.

$$d_i = \sqrt{(|x_i - x_{i+1}|)^2 + (|y_i - y_{i+1}|)^2} \qquad (2)$$

d_i merupakan vektor kumpulan jarak dari 2 titik yang berdekatan, dimulai dari awal bergerakan hingga *task* klik dimulai.

```
513,1166,463,705578390
514,1166,463,705578500
512,1166,463,705578765 -> awal gerakan (x & y awal)
...
512,91,732,705580109
512,91,732,705580140    -> akhir gerakan (x & y akhir)
513,91,732,705580234    -> kode aksi click down
```

Gambar 3.13. *Raw data* hasil tangkapan, menunjukkan *movement task* di antara *click task*

Pada Gambar 3.13, data pergerakan tetikus dengan kode 512 merupakan kode *movement* yang berarti posisi tetikus di saat melakukan perpindahan posisi. Data ini ditangkap dengan interval 16 milisekon. Untuk menghitung nilai jarak *offset* penulis perlu menghitung jarak antar 2 baris data dengan kode 512 yang terdapat di antara kode 514 dan 513. Contoh data dalam bentuk grafik dapat dilihat pada Gambar 3.14.

Gambar 3.14. Grafik menunjukkan koordinat pergerakan tetikus.

Pada Gambar 3.14, di titik (421, 571) dan (506,524) terdapat segitiga hijau, sebagai penanda. Untuk mendapatkan jarak antar dua titik tersebut digunakan jarak Euclidian. Jumlah data untuk tiap jarak euclidian yang didapat dalam satu gerakan tidaklah tetap. Hal ini tidak menjadi masalah karena data yang diambil adalah selisih dari jarak optimum d_{opt} dengan jumlah jarak d_i. Garis merah pada Gambar 3.14

menunjukkan jarak optimum dalam satu pergerakan, diambil dari jarak euclidian antara titik awal dan akhir (kode 514 hingga kode 513).

d) Waktu tempuh

Waktu tempuh atau *Movement elapsed time* didapatkan dengan menghitung total waktu yang dibutuhkan oleh user untuk setiap *task* gerakan. Total ada 16 data *movement elapsed time*

512,412,275,705575484 ← awal gerak

...

512,707,313,705576781 ← akhir gerak

Gambar 3.15. *Movement elapsed time* data

Pada Gambar 3.15, data dengan kode 512 pada awal rentetan data menjadi titik awal, dan data kode 512 yang ada sebelum kode 513 pada rentetan data menjadi titik akhir, nilai di kolom empat (data paling kanan) merupakan data timestamp. Dengan mencari selisih timestamp titik awal dan akhir penulis mendapatkan nilai *movement elapsed time*.

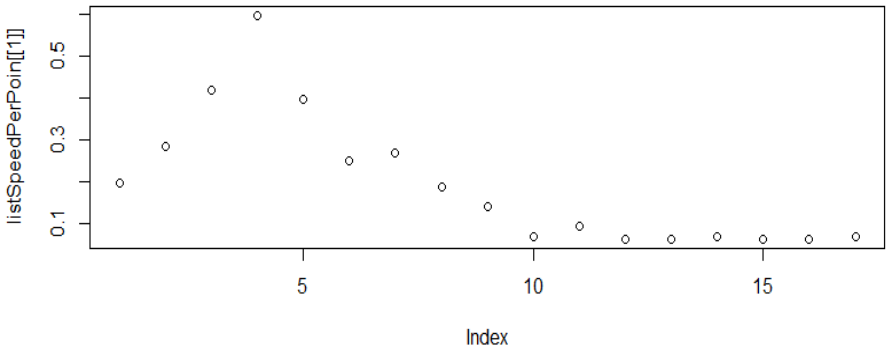
2. Fitur Prosedural

a) Kecepatan

Kecepatan didapatkan dengan menghitung kecepatan antar 2 titik. Data ini merupakan data *time-seies* yang disimpan dalam format vektor. Setiap *task* menghasilkan satu vektor dengan isi data yang jumlahnya dinamis. Total ada 16 vektor data kecepatan

```
> listSpeedPerPoin[1]
[[1]]
 [1] 0.19764235 0.28284271 0.41926275 0.59628479 0.39528471
 [6] 0.25000000 0.26666667 0.18750000 0.13975425 0.06666667
[11] 0.09428090 0.06250000 0.06250000 0.06666667 0.06250000
[16] 0.06250000 0.06666667
```

Gambar 3.16. *List* nilai kecepatan.



Gambar 3.17. Contoh kurva kecepatan berdasarkan Gambar 3.16.

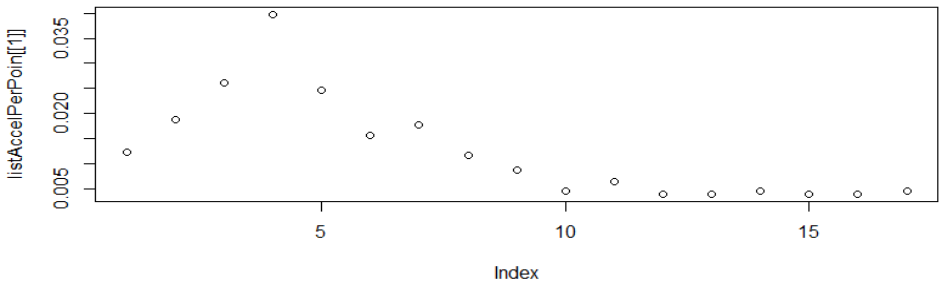
Pada Gambar 3.16, data kecepatan disimpan dalam objek vector bernama *listspeedperpoint*. Didalamnya terdapat 17 nilai kecepatan. Sebagai analisis, pada Gambar 3.17, merupakan kurva dari data kecepatan *listspeedperpoint* , dimana pola yang terbentuk akan berbeda tiap orang.

b) Percepatan

Percepatan atau akselerasi didapatkan dengan menghitung percepatan antar 2 titik. Data disimpan dalam format vektor. Setiap *task* menghasilkan satu vektor dengan isi data yang jumlahnya dinamis. Total ada 16 vektor data percepatan.

```
> listAccelPerPoin[1]
[[1]]
 [1] 0.012352647 0.018856181 0.026203922 0.039752320
 [5] 0.024705294 0.015625000 0.017777778 0.011718750
 [9] 0.008734641 0.004444444 0.006285394 0.003906250
[13] 0.003906250 0.004444444 0.003906250 0.003906250
[17] 0.004444444
```

Gambar 3.18. Contoh data akselerasi.



Gambar 3.19. Kurva akselerasi.

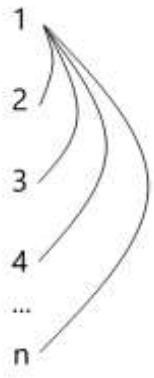
Pada Gambar 3.18, data akselerasi disimpan dalam objek vector bernama *listsAccelPerPoint*. Didalamnya terdapat 17 nilai akselerasi. Sebagai analisis, pada Gambar 3.19, merupakan kurva dari data akselerasi kecepatan *listsAccelPerPoint*, dimana pola yang terbentuk akan berbeda tiap orang.

3.5. *Distance Measurement*

Data Fitur hasil ekstraksi belum bisa digunakan sebagai input *classifier*. Diperlukan perhitungan jarak antar vektor fitur untuk menemukan area pola dari koordinat fitur. Dalam penelitian ini digunakan 2 formula perhitungan jarak yaitu *Manhattan distance* dan *dynamic time wrapping distance* (DTW).

Pada penelitian ini data terbagi menjadi fitur holistik dan fitur prosedural dimana holistik adalah data statistik satu dimensi (waktu & jarak) sementara prosedural adalah data vektor berupa kecepatan dan percepatan. Fitur holistik berjumlah 42 fitur sementara fitur prosedural berjumlah 32 fitur. Fitur holistik memiliki nilai tunggal sementara fitur prosedural berupa vektor *time-series.*

Data dihitung sesuai dengan kategori masing-masing. Setiap baris dipasangkan dengan baris lainnya sehingga nilai *distance* yang didapatkan berjumlah n(n-1) buah vektor.



Gambar 3.20. Ilustrasi pasangan data untuk menghitung jarak.

Pada Gambar 3.20 , angka menunjukkan nomor baris dan *n* adalah total baris pada dataset. Untuk setiap baris data akan dipasangkan dengan data yang lain dalam dataset sehingga setiap baris akan memiliki n-1 data distance. Jumlah data distance yang terbentuk berjumlah n*(n-1) dimana n merupakan jumlah baris data.

Penelitian ini menggunakan *Manhattan distance* sebagai algoritma untuk menghitung jarak antar fitur holistik, dan *Dynamice Time Warping* (DTW) distance untuk menghitung jarak antar vektor fitur prosedural.

3.5.1. *Manhattan Distance*

Metode perhitungan ini digunakan pada fitur holistik. Metode ini menghitung jarak antar dua nilai tunggal yang hasilnya berupa nilai positif.

$$d_{i,j}^{H} = \left| x_i^H - x_j^H \right| \tag{3}$$

Misal data fitur holistik sebagai berikut,

Tabel 3.4. Contoh 3 observasi fitur holistik.

No. Observasi	Fitur							
	f1	f2	f3	f4	f5	f6	...	f42
O1	89.875	11.06394	75.54167	17.00123	84	58.75	...	297
O2	87.75	14.23025	70.29167	14.53176	74.25	56.5	...	391
O3	74.25	16.49892	76.91667	20.6143	78.25	54.75	...	281

Dihitung jarak antara observasi 1 dan observasi 2 untuk kolom pertama maka,

$$d^H = |v1_1 - v2_1|$$
$$d^H = |82 - 84.125|$$
$$d^H = 2.125$$

Perhitungan diulangi untuk semua kolom hingga kolom ke 42 dalam pasangan tersebut. Kemudian perhitungan diulangi untuk seluruh pasangan yang mungkin, sehingga dihasilkan n(n-1) buah data. Pada contoh 3 observasi, didaptkan 6 buah data *distance* dengan hasil seperti berikut.

Tabel 3.5. *Pairwise distance* fitur holistik 3 observasi.

No. distance	Fitur							
	f1	f2	f3	f4	f5	f6	...	f42
$d_{1,2}$	2.125	3.166311	5.25	2.469463	9.75	2.25	...	94
$d_{1,3}$	15.625	5.43498	1.375	3.613073	5.75	4	...	16
$d_{2,1}$	2.125	3.166311	5.25	2.469463	9.75	2.25	...	94
$d_{2,3}$	13.5	2.268669	6.625	6.082536	4	1.75	...	110
$D_{3,1}$	15.625	5.43498	1.375	3.613073	5.75	4	...	16
$d_{3,2}$	13.5	2.268669	6.625	6.082536	4	1.75	...	110

Distance antara observasi 1 dan observasi 2 dilambangkan dengan $d_{1,2}$, seterusnya bagi semua pasangan. perhitungan diulangi dengan pasangan yang berbeda sehingga $d_{n,n-1}$. Distance vektor dengan dirinya sendiri tidak dihitung.

3.5.2. *Dynamic Time Warping Distance*

Metode perhitungan ini digunakan pada fitur prosedural. Metode ini menghitung jarak antar *timeseries* dengan panjang yang berbeda. Hasil dari perhitungan DTW adalah nilai tunggal positif.

$$d_{i,j}^P = DTW(x_i^P, x_i^P) \quad (4)$$

Misal, vektor hasil ekstraksi fitur sebagai berikut

Tabel 3.6. Contoh fitur prosedural.

No.observasi	Fitur kecepatan

	f1	f2	...	f32
O1	0.22534695	0.2576941		1.099611882
	0.28284271	0.6		2.911383512
	0.03225806	1.37641972		3.499546824
	0.0625	2.60085456		4.777977867
	0.08064516	3.4375		4.336274622
	0.50389111	5.56285111		1.955016598
	0.8	4.53333333		1.923915808
	1.0097184	1.75		0.545348659
	0.6731456	0.8125		0.841636562
	0.86666667	0.06666667	...	0.271992581
O2	0.1490712	0.625		0.818724208
	0.0625	2.06666667		2.463647056
	0.19764235	3.375		4.09142743
	0.18856181	4.00055552		4.699426292
	0.08838835	3.82068793		4.236821209
	0.13975425	1.87604138		2.455627517
	0.35901099	0.93571126		1.51384338
	0.44194174	0.2576941		0.86560343
	0.88944427	0.125		0.361619718
	0.69877124	0.06666667	...	0.481238836

Timeseries1 dijadikan sebagai sumbu x, dan timeseries2 dijadikan sumbu y. Sehingga terlihat seperti gambar

Tabel 3.7. *Distance matrix* DTW.

		O2 (reference)									
no		1	2	3	4	5	6	7	8	9	10
O1 (query)	nilai	0.149071	0.0625	0.197642	0.188562	0.088388	0.139754	0.359011	0.441942	0.889444	0.698771
1	0.225347	0.076276	0.239123	0.266827	0.303613	0.440571	0.526164	0.659828	0.876423	1.54052	2.013944
2	0.282843	0.210047	0.43039	0.352028	0.397893	0.592348	0.669252	0.6785	0.837599	1.444201	1.860129
3	0.032258	0.32686	0.270531	0.435915	0.554197	0.510154	0.61765	0.944403	1.247283	2.104469	2.526643
4	0.0625	0.413432	0.270531	0.405674	0.531735	0.536042	0.613297	0.909808	1.289249	2.116194	2.752465
5	0.080645	0.481858	0.288676	0.405674	0.51359	0.521333	0.580442	0.858808	1.220105	2.028904	2.64703
6	0.503891	0.836678	0.730067	0.711922	0.828919	0.936836	0.944579	0.870203	0.932152	1.317705	1.512585
7	0.8	1.487606	1.467567	1.31428	1.440358	1.648448	1.604825	1.311192	1.29021	1.111041	1.212269
8	1.009718	2.348254	2.414786	2.126356	2.261514	2.569778	2.474789	1.961899	1.857987	1.231315	1.523216
9	0.673146	2.872328	3.025431	2.601859	2.746098	3.154535	3.008181	2.276034	2.089191	1.447613	1.282566

10	0.866667	3.589923	3.829598	3.270884	3.424203	3.932813	3.735093	2.783689	2.513916	1.470391	1.450461

Dihitung dengan cara :

$D = |A_1 - B_1|$ untuk data paling pertama; kotak merah.

$D = |A_i - B_j| + D[i-1, 0]$ untuk baris data berwana oranye.

$D = |A_i - B_j| + D[0, j-1]$ untuk baris data berwarna kuning.

$D = |A_i - B_j| + \min(D[i-1, j-1], D[i-1, j], D[i, j-1])$ untuk data lainnya.

DTW *distance* yang dipilih ditandai dengan angka berwarna merah. Dihitung dengan cara mencari nilai paling minimum diantara tiga kotak di sekeliling kotak paling kanan atas, menuju ke kiri bawah. Persamaan untuk memilih langkah DTW adalah

```
g[i,j] = min(
     g[i-1,j-1] + 2 * d[i  ,j  ] ,
     g[i  ,j-1] +     d[i  ,j  ] ,
     g[i-1,j  ] +     d[i  ,j  ] ,
)
```

Global distance dari vektor A dan vektor B adalah 1.450461 , yang merupakan jarak DTW antara 2 vektor timeseries tersebut.

3.5.3. *Vektor referensi*

Data yang telah digabung kemudian di normalisasi dengan z-factor untuk mendapatkan data dengan skala yang sama. Hal iini dilakukan karena data-data hasil *distance measurment* belum tentu memiliki skala yang sama, dan untuk memaksimalkan varian data. Selanjutnya perlu ditentukan referensi vektor fitur , dimana vektor ini dijadikan vektor basis untuk menghitung jarak fitur , holistik maupun prosedural. Misal x_{ref} adalah vektor referensi.

$$x_{ref} = x_k, k = arg \min_i \frac{1}{k-1} \sum_{j=1, j \neq i}^{n} \sqrt{\sum_{l=1}^{d} \left(\bar{d}_{i,j}^{l}\right)^2} \quad (5)$$

Vektor referensi di ambil dari vektor yang mempunyai nilai mean jarak paling rendah dari semua pasangan vektor yang ada. Dimana *k* adalah nomor

baris dengan mean jarak paling kecil. Pasangan yang dimaksud adalah pasangan perhitungan jarak yang telah dilakukan, misal ada 3 buah observasi, maka dihasilkan 3 pasang jarak masing-masing berisi 2 vektor jarak. Diantaranya vektor jarak observasi 1 & 2, dan observasi 1 & 3 . inilah yang disebut 1 pasang. Setiap nilai pada fitur diubah ke dalam bentuk positif dan dijumlahkan, lalu hasil penjumlahan fitur dijumlahkan dengan hasil dari vektor lain dalam pasangan dan dihitung meannya, setelah didapat semua mean dari masing-masing pasangan, dipilih mean yang terkecil untuk dijadikan vektor referensi. Setiap subjek akan memiliki satu buah vektor referensi. Misal data mean dari jarak pasangan berikut

Tabel 3.8. *Pairwise distance.*

No. Distance	Fitur							
	f1	f2	f3	f4	f5	f6	...	f74
$d_{1,2}$	2.125	3.166311	5.25	2.469463	9.75	2.25	...	94
$d_{1,3}$	15.625	5.43498	1.375	3.613073	5.75	4	...	16
$d_{2,1}$	2.125	3.166311	5.25	2.469463	9.75	2.25	...	94
$d_{2,3}$	13.5	2.268669	6.625	6.082536	4	1.75	...	110
$d_{3,1}$	15.625	5.43498	1.375	3.613073	5.75	4	...	16
$d_{3,2}$	13.5	2.268669	6.625	6.082536	4	1.75	...	110

Sebagai contoh warna oranye menunjukkan pasangan *distance* observasi 1. Pertama-tama dihitung jumlah nilai seluruh fitur pada tiap vektor. Untuk vektor $d_{1,2}$ dan $d_{1,3}$:

$$m = \sqrt{f1^2 + f2^2 + \cdots + f74^2}$$

Didapatkan dua nilai 2 nilai m , kemudian dirata-rata per pasangan. Hasil perhitungan ada pada Tabel 3.9.

Tabel 3.9. Contoh mean *distance* per pasangan (*pairwise distance*).

no. *Distance*	Fitur						mean per pasangan
	f1	f2	f3	...	f74	*m*	
$d_{1,2}$	-1.39898	-0.34299	0.374898	...	0.503284	2.309003	2.437746
$d_{1,3}$	0.878758	1.359674	-1.36838	...	-1.39621	2.566489	

$d_{2,1}$	-1.39898	-0.34299	0.374898	...	0.503284	2.309003	2.38755
$d_{2,3}$	0.520225	-1.01668	0.993478	...	0.892923	2.466097	
$d_{3,1}$	0.878758	1.359674	-1.36838	...	-1.39621	2.566489	2.516293
$d_{3,2}$	0.520225	-1.01668	0.993478	...	0.892923	2.466097	

Dapat dilihat pada Tabel 3.11, dari 6 *pairwise distance*, pasangan dengan nilai mean paling rendah adalah $d_{2,1}, d_{2,3}$. Maka k dalam $x_{ref} = x_k$ adalah nomor indeks dari pasangan (*pairwise*) *distance* $d_{2,1}, d_{2,3}$ yaitu indeks data observasi 2 (O2). Dengan begitu vektor O2 akan menjadi vektor referensi untuk perhitungan *distance* data baru.

3.6. *Principal Component Analysis*

Penggunaan fitur jarak secara langsung belum cukup untuk mendapatkan nilai akurasi yang tinggi. Diperlukan metode *dimensionality reduction* yang bisa memilah dan mengurangi dimensi data agar hasil lebih akurat. Pada penelitian ini digunakan *principal component analysis* untuk mengurangi dimensi data yang diharapkan mampu meningkatkan akurasi sistem dan mempercepat proses autentikasi.

Data fitur vektor yang dihasilkan mengandung 74 dimensi dimana 74 dimensi ini pasti memilki data yang tingkat kepentingannya lebih tinggi dari yang lain. Menggunakan seluruh fitur juga tidak terlalu diharapkan karena dapat memperlama proses autentikasi. Penulis akan menggunakan berbagai jumlah komponen dari 20, 30, 40, 50 dan 60.

Misal data dengan 3 fitur, sebagai berikut

Tabel 3.10. Tabel data contoh input PCA

	X1	X2	X3
1	5.1	3.5	1.4
2	4.9	3	1.4
3	4.7	3.2	1.3
4	4.6	3.1	1.5
5	5	3.6	1.4

Standardisasikan data dengan z-score, sehingga menjadi seperti Tabel 3.11

Tabel 3.11. Matriks standardisasi.

	x1	x2	x3
1	1.293993	0.950255	0
2	0.215666	-1.20942	0
3	-0.86266	-0.34555	-1.58114
4	-1.40183	-0.77748	1.581139
5	0.754829	1.382189	0

Kemudian dihitung matriks kovarian, sehingga menghasilkan data seperti Tabel 3.12

Tabel 3.12. Matriks kovarian

	x1	x2	x3
1	1	0.680019	-0.1705
2	0.680019	1	-0.13659
3	-0.1705	-0.13659	1

Setelah itu dilakukan perhitungan *eigendecomposition* sehingga menghasilkan pasangan eigenvalue dan eigenvector seperti Tabel 3.13

Tabel 3.13. Hasil *eigenvalue* dan *eigenvector*

Eigenvalue	e1	e2	e3
	1.74347	0.937435	0.319091
Eigenvector	v1	v2	v3
	-0.68205	0.171985	-0.71079
	-0.67537	0.224658	0.702425
	0.280491	0.95914	-0.03708

Proyeksikan data orisinal ke ruang pca . menggunakan eigenvector sesuai urutan fiturnya. Hasil pryoyeksi pada Tabel 3.16.

Tabel 3.14. Matriks hasil proyeksi ke ruang PCA.

	PC1	PC2	PC3
1	-1.36342	0.389996	-0.22564
2	0.599007	-0.20985	-0.89695
3	0.338326	-1.55856	0.383774

4	1.721509	0.984562	0.350313
5	-1.29542	0.393851	0.388502

Hasil proyeksi bisa direduksi ke 2 dimensi dengan cara menghilangkan vektor dengan eigenvalue paling kecil, sehingga *principal component* yang dihasilkan adalah PC1 dan PC2 saja.

3.7. *Kernelized Oneclass Support Vector Machine*

Pada penelitian yang berkaitan dengan sistem autentikasi, metode klasifikasi *novelty detection* menjadi cara yang ampuh untuk mengelompokkan data dari pengguna valid dan tidak valid. Karena pada penelitian ini hanya menggunakan sampel dari pengguna valid (sampel positif) digunakanlah *oneclass support vector machine* (OC-SVM), yang mampu mengolompokkan data hanya dengan satu kelompok sampel data.

Misal data input hasil PCA sebagai berikut,

Tabel 3.15. Input SVM.

	PC1	PC2	PC3
1	-1.36342	0.389996	-0.22564
2	0.599007	-0.20985	-0.89695
3	0.338326	-1.55856	0.383774
4	1.721509	0.984562	0.350313
..	...	...	...
5550	-1.29542	0.393851	0.388502

Setelah melalui proses SVM, dihasilkan *support vector* dengan dimensi sejumlah fitur yang ada. Jumlah *support vector* tergantung setting variabel gamma dan nu.

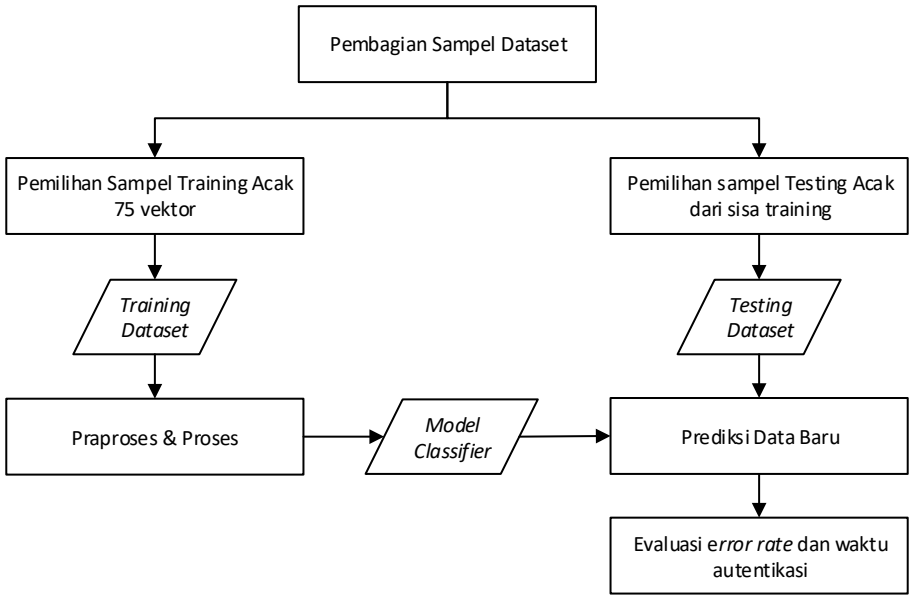
Tabel 3.16. Output SVM.

No. SV	PC1	PC2	PC3
1	-0.21933	0.345117	2.444877
2	-0.92135	-2.22834	0.314808
3	0.264339	0.346345	0.589635
4	-0.07522	-0.67859	2.134438
..	...	...	...

54	-1.09841	-0.7062	2.017735

3.8. Desain Eksperimen

Pada penelitian ini dilakukan beberapa langkah dalam melakukan eksperimen yang dilakukan seperti Gambar 3.21.



Gambar 3.21. Desain eksperimen.

Tahapan eksperimen yang dilakukan seperti terpapar pada Gambar 3.21 adalah sebagai berikut:

1. Setiap dataset yang dikumpulkan akan dibagi menjadi data *training* dan data *testing*. Data training yang diambil berjumlah 75 data per subjek dan sisa data digunakan sebagai data *testing*.
2. Penulis melakukan *training* pada data *training* yang diambil secara acak pada setiap percobaan.
3. Penulis menguji kemampuan *classifier* untuk mengenali pengguna valid dengan menghitung skor anomali dari data *testing*. Semakin kecil nilai ini maka semakin baik. Penulis menyebut skor ini sebagai *genuine score*.

4. Penulis menguji kemapuan *classifier* untuk mengenali peniru (pengguna tidak valid) dengan menghitung skor anomali yang dihasilkan oleh peniru. Penulis menyebut skor ini sebagai *impostor score*.

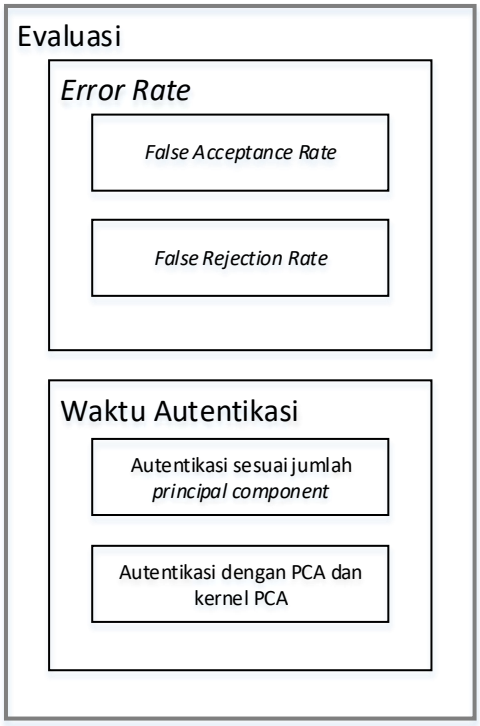
Proses ini diulang, menunjuk tiap subjek yang ada sebagai pengguna valid secara bergantian. Karena penulis menggunakan teknik sampel acak maka proses tersebut diulang sebanyak 50 kali, setiap perulangan menggunakan data *training* acak yang baru dari keseluruhan data set.

Berikut lingkungan sistem yang digunakan dalam penelitian ini.

1. *Hardware*
   - Processor a8 3870k 4 Core @3GHz
   - 8GB DDR3 RAM
   - AMD RX460 2GB GDDR5
   - 2 TB HDD 5400 RPM
2. *Software*
   - Windows 10 pro 64bit
   - R 3.4.4
   - Excel 2013
   - Mouse capture software (c#)

3.9. Evaluasi

Untuk melakukan evaluasi sistem autentikasi dengan biometrik tetikus diperlukan serangkaian proses diantaranya evaluasi dengan *False Acceptance Rate*, *False Rejection Rate,* dan waktu autentikasi (Ahmed & Traore, 2007). Evaluasi yang dilakukan diilustrasikan pada Gambar 3.22.



Gambar 3.22. Ilustrasi Evaluasi.

Dengan demikian, peneliti memutuskan untuk membuat skenario evaluasi dengan mempertimbangkan 2 aspek, yaitu:

1. *Error Rate*, yaitu mengevaluasi tingkat galat dari suatu sistem autentikasi biometrik, yang termasuk didalam *error rate* ini adalah *False Acceptance Rate (FAR)* dan *False Rejection Rate (FRR)*. Peneliti memutuskan untuk menggunakan teknik tersebut dengan skenario berikut: setiap subjek akan di test dengan data testing dari pengguna valid secara acak sejumlah dengan data test dari pengguna lain sebagai pengguna tidak valid. Kemudian hasil akurasi dari setiap kategori diambil nilai rata-ratanya.
2. Waktu autentikasi, yaitu mengevaluasi waktu komputasi sistem saat melakukan autentikasi data baru menggunakan perintah System.time(). Perhitungan dilakukan untuk semua hasil autentikasi yang dilakukan pada setiap kategori (FAR & FRR) kemudian dihitung rata-ratanya.

Selain menilai kedua aspek tersebut, peneliti juga melakukan perbandingan fitur dengan penelitian-penelitian terkait dan uji coba menggunakan *neural network* sebagai pembanding.