BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Pada bab ini dilakukan impementasi dan pengujian terhadap sistem yang menggunakan metode SIFT dan KNN. Hasil impementasi akan di uji kebenarannya melalui tahapan-tahapan pengujian yang telah ditentukan. Tahapan ini dilakukan setelah perancangan selesai dilakukan selanjutnya akan di implementasikan kedalam bahasa pemograman.

4.1.1 Batasan Implementasi

Batasan implementasi dimaksudkan agar ruang lingkup dari implementasi tidak terlalu luas dan implementasi jelas. Pembatasan implementasi dari sistem adalah sebagai berikut :

1. Perangkat lunak yang digunakan berbasis desktop.
2. Sistem operasi yang digunakan adalah Windows 10 64 bit.
3. Untuk implementasinya dibutuhkan sebuah data dari *frame* WebCam sebanyak 7 jenis *gesture* tangan dan berjumlah 20 citra pada setiap jenis *gesture* tangan.
4. Dalam implementasi, citra *gesture* tangan diambil dari hasil pemotongan *frame* sebesar 200 x 200 dari *webcam* secara *realtime*.

4.1.2 Implementasi Antarmuka

Dari perancangan antarmuka yang telah dibuat pada bab sebelumnya, maka tahap selanjutnya yaitu mengimplementasikan menjadi sebuah tampilan. Implementasi antarmuka sistem terdapat pada Tabel 4.1 yaitu sebagai berikut :

Tabel 4.1 Implementasi Antarmuka

No	Nama Antarmuka	Deskripsi
1	Halaman Utama	Ketika pengguna membuka aplikasi pertama kali, maka antarmuka yang akan ditampilkan adalah antarmuka Halaman Utama

2	Tombol *Preprocessing*	Ketika pengguna menekan tombol *preprocessing*, maka yang akan ditampilkan adalah hasil preprocessing pada citra.
3	Tombol Ekstraksi Ciri (SIFT)	Ketika pengguna menekan tombol ekstraksi ciri (SIFT), maka yang akan ditampilkan adalah hasil ekstrasi ciri pada citra.
4	Tombol Capture Gambar Training	Ketika pengguna menekan tombol capture gambar training, maka yang akan ditampilkan adalah popup jenis data training. Kemudian pengguna memilih jenis data training yang akan disimpan.
5	Tombol Training Data (KNN)	Ketika pengguna menekan tombol training data (KNN), maka sistem akan melakukan training data menggunakan metode KNN.
6	Tombol Mulai Testing	Ketika pengguna menekan tombol mulai testing, maka sistem akan melakukan testing data dan menampilkan hasilnya pada layer.

4.1.3 Implementasi Library

Untuk bisa membangun sistem akan digunakan *library* Open CV 3.3.0 + OpenCV Contrib dengan bahasa pemrograman java. Dalam penggunaan *library* Open CV 3.3.0 dengan menggunakan bahasa pemrograman java, diperlukan beberapa *library* yang di *import* untuk menghubungkan antara sistem yang akan dibangun dengan *library* openCV. *Library* tersebut adalah sebagai berikut :

1. *Import org.opencv.core.Core;*
2. *Import org.opencv.core.CvType;*
3. *Import org.opencv.core.Mat;*
4. *Import org.opencv.core.MatOfFloat;*
5. *Import org.opencv.Size;*
6. *Import org.opencv.TermCriteria;*
7. *Import org.opencv.highgui.Highgui;*

8. *Import org.opencv.imgproc.Imgproc;*
9. *Import org.opencv.ml.KNearest;*
10. *Import org.opencv.xfeatures2d.SIFT;*

Berikut ini merupakan implementasi *library* dari tiap – tiap metode yang digunakan seagai berikut :

1. Mengubah RGB ke HSV

Dalam Open CV Java, metode yang digunakan untuk konversi citra RGB ke HSV adalah sebagai berikut :

```
Imgproc.cvtColor(Mat src, Mat dst, int code);
```

Penjelasan :

a. Library yang harus di import adalah import *org.opencv.imgproc.Imgproc*;
b. `Mat src` merupakan sumber atau citra awal sebagai masukkan untuk metode ini.
c. `Mat dst` merupakan destinasi dari hasil citra HSV sebagai keluaran untuk metode ini.
d. `int code` merupakan kode untuk pengaturan pengubahan warna skala keabuan yang dimana pada penelitian ini kode yang digunakan adalah `Imgproc.COLOR_BGR2GRAY`.

2. Mengambil nilai HSV hanya pada warna merah (*Thresholding*)

Dalam Open CV Java, metode yang digunakan untuk mengambil nilai HSV hanya pada warna merah adalah sebagai berikut :

```
Core.inRange(Mat src, Scalar lowerb, Scalar upperb, Mat dst);
```

Penjelasan :

a. Library yang harus di import adalah *import org.opencv.core.Core*;
b. `Mat src` merupakan sumber atau citra awal sebagai masukkan untuk metode ini.

c. `Mat dst` merupakan destinasi dari hasil citra *thresholding* sebagai keluaran untuk metode ini.
d. `Scalar lowerb` merupakan nilai batas bawah untuk nilai HSV warna merah.
e. `Scalar lowerb` merupakan nilai batas atas untuk nilai HSV warna merah.

3. Melakukan erosi dan dilasi (*Opening*)

Dalam Open CV Java, metode yang digunakan untuk erosi dan dilasi adalah sebagai berikut :

```
Imgproc.erode(Mat src, Mat dst, Mat kernel);
Imgproc.dilate(Mat src, Mat dst, Mat kernel);
```

Penjelasan :

a. Library yang harus di import adalah *org.opencv.imgproc.Imgproc*;
b. `Mat src` merupakan sumber atau citra awal sebagai masukkan untuk metode ini.
c. `Mat dst` merupakan destinasi dari hasil citra erosi atau dilasi sebagai keluaran untuk metode ini.
d. `Mat kernel` merupakan nilai kernel yang digunakan untuk proses erosi dan dilasi.

4. Melakukan dilasi dan erosi (*Closing*)

Dalam Open CV Java, metode yang digunakan untuk dilasi dan erosi adalah sebagai berikut :

```
Imgproc.dilate(Mat src, Mat dst, Mat kernel);
Imgproc.erode(Mat src, Mat dst, Mat kernel);
```

Penjelasan :

a. Library yang harus di import adalah *org.opencv.imgproc.Imgproc*;
b. `Mat src` merupakan sumber atau citra awal sebagai masukkan untuk metode ini.

c. `Mat dst` merupakan destinasi dari hasil citra erosi atau dilasi sebagai keluaran untuk metode ini.
d. `Mat kernel` merupakan nilai kernel yang digunakan untuk proses erosi dan dilasi.

5. Menemukan kontur terbesar tangan

Dalam Open CV Java, metode yang digunakan untuk menemukan kontur terbesar tangan adalah sebagai berikut :

```
Imgproc.findContours(Mat image, List<MatOfPoint> contours, Mat hierarchy, int mode, int method);
```

Penjelasan :

a. Library yang harus di import adalah *org.opencv.imgproc.Imgproc*;
b. `Mat image` merupakan sumber atau citra awal sebagai masukkan untuk metode ini.
c. `List<MatOfPoint> contours` merupakan nilai kontur tangan yang terdeteksi.
d. `Mat hierarchy` merupakan nilai keluaran vektor dan menggambarkan topologi gambar berdasarkan *contours*.
e. `int mode` merupakan kode untuk pengaturan nilai *contours*.
f. `int method` merupakan kode untuk pengaturan *approximate contours*.

6. Mendapatkan ROI tangan

Dalam Open CV Java, metode yang digunakan untuk mendapatkan ROI tangan adalah sebagai berikut :

```
Imgproc.rectangle(Mat img, Point point1, Point point2, Scalar color, int thickness);
```

Penjelasan :

a. Library yang harus di import adalah *org.opencv.imgproc.Imgproc*;
b. `Mat img` merupakan sumber atau citra awal sebagai masukkan untuk metode ini.

c. `Point point1` merupakan batas atas kontur tangan.
d. `Point point2` merupakan batas bawah kontur tangan.
e. `Scalar color` merupakan warna kotak yang diinginkan untuk menggambar ROI tangan.
f. `int thickness` merupakan ketebalan garis ROI tangan.

7. Konversi citra RGB menjadi citra Grayscale

Dalam Open CV java, metode yang digunakan untuk mengkonvcersi citra RGB menjadi citra Grayscale adalah sebagai berikut :

```
Imgproc.cvtColor(Mat src, Mat dst, int code);
```

Penjelasan :

a. Library yang harus di import adalah import *org.opencv.imgproc.Imgproc*;
b. `Mat src` merupakan sumber atau citra awal sebagai masukkan untuk metode ini.
c. `Mat dst` Merupakan destinasi dari hasil citra Grayscale sebagai keluaran untuk metode ini.
d. `int code` merupakan kode untuk pengaturan pengubahan warna skala keabuan yang dimana pada penelitian ini kode yang digunakan adalah `Imgproc.COLOR_BGR2GRAY`.

8. Melakukan Ekstrasi Ciri SIFT

Dalam openCV java, metode yang digunakan untuk melakukan ekstrasi ciri SIFT adalah sebagai berikut :

```
SIFT d = SIFT.create();
d.detectAndCompute(Mat img, new Mat(), MatOfKeyPoint kp, MatOfKeypoint dp);
```

Penjelasan :

a. Library yang harus di import adalah *import org.opencv.xfeatures2d.SIFT* dan *import org.opencv.features2d.Features2d;*

b. `SIFT.create()` merupakan metode untuk melakukan ekstrasi ciri SIFT.
c. `Mat img` merupakan sumber atau citra awal sebagai masukkan untuk metode ini.
d. `MatOfKeyPoint kp` merupakan *keypoint* yang ditemukan pada citra gambar.
e. `MatOfKeypoint dp` merupakan nilai fitur vektor SIFT yang disimpan dalam bentuk matriks.

9. Melakukan KNN

Dalam Open CV java, metode yang digunakan untuk melakukan KNN adalah sebagai berikut :

```
private KNearest knn = KNearest.create();
knn.train(Mat trainData, Ml.ROW_SAMPLE, Mat trainLabel);
knn.save(XML);
knn.findNearest(Mat dp, int k, Mat result);
```

Penjelasan :

a. Library yang harus di import adalah *import org.opencv.ml.KNearest* dan *import org.opencv.ml.Ml*;
b. `KNearest.create()` merupakan metode untuk melakukan KNN.
c. `Mat trainData` merupakan kumpulan nilai vektor SIFT.
d. `Mat trainLabel` merupakan kumpulan label klasifiikasi.
e. `Mat dp` merupakan vektor SIFT yang menjadi data testing.
f. `int K` merupakan nilai K tettangga terdekat untuk melakukan KNN.
g. `Mat resilt` merupakan hasil keluaran dari pencarian K terdekat

4.2 Pengujian Perangkat Lunak

Pengujian yang dilakukan adalah *blackbox*. Pada pengujian *blackbox testing* yaitu menemukan kesalahan yang terdapat pada program.

4.2.1 Pengujian *Whitebox*

Pengujian *whitebox* digunakan untuk menguji performa metode yang digunakan. Berikut adalah kasus menguji perangkat lunak yang telah dibangun menggunakan metode *whitebox*.

4.2.1.1 Pengujian Melakukan *Streaming Webcam*

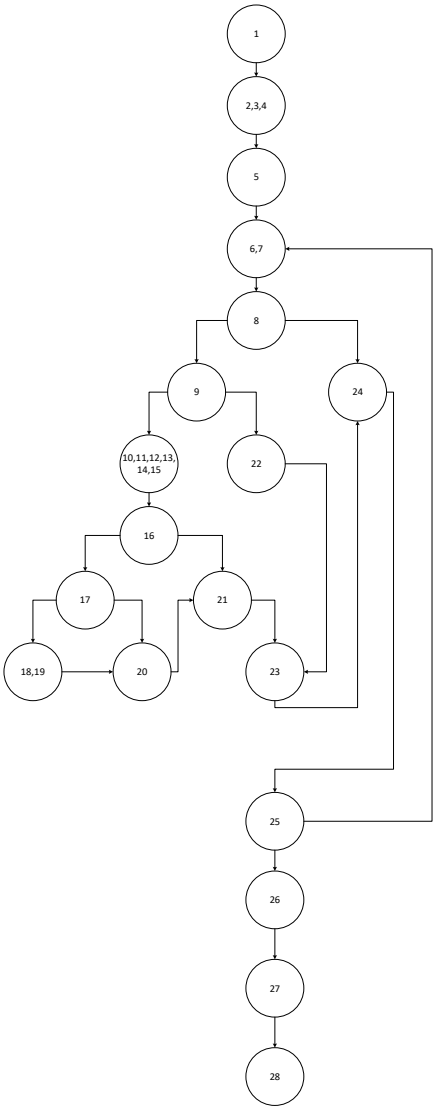
Pengujian melakukan *streaming webcam* dapat dilihat pada tabel 4.2

Tabel 4.2 Melakukan *streaming webcam*

Line	Source
1	`class DaemonThread implements Runnable {`
2	`protected volatile boolean runnable = false;`
3	`@Override`
4	`public void run() {`
5	`synchronized (this) {`
6	`trainKNN();`
7	`while (runnable) {`
8	`if (webSource.grab()) {`
9	`try {`
10	`webSource.retrieve(citra);`
11	`Imgcodecs.imencode(".bmp", citra, mem);`
12	`Image im = ImageIO.read(new ByteArrayInputStream(mem.toArray()));`
13	`BufferedImage buff = (BufferedImage) im;`
14	`Mat citraAsli = citra;`
15	`Graphics g = panelCamera.getGraphics();`
16	`if (g.drawImage(citraAsli, 0, 0, 640, 480, null)) {`
17	`if (runnable == false) {`
18	`System.out.println("Going to wait()");`
19	`this.wait();`
20	`}`
21	`}`
	`} catch (Exception ex) {`
22	`System.out.println("Error");`
23	`}`
24	`}`
25	`}`
26	`}`
27	`}`
28	`}`

1. *Flow Graph* Melakukan *Streaming Webcam*

Berdasarkan tabel 4.2 maka dapat dibentuk *flow graph* seperti pada gambar 4.1.



Gambar 4.1 *Flow Graph* Melakukan *Streaming Webcam*

Dari gambar 4.1 dapat dihitung *cylomatic complexity* sebagai berikut :

V(G) = E – N + 2

V(G) = 23 – 18 + 2 = 5

Jadi, *cyclomatic complexity* untuk gambar 4.1 adalah 5. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 5 *path* yang terdiri dari:

Path 1 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-23-24-25-26-27-28

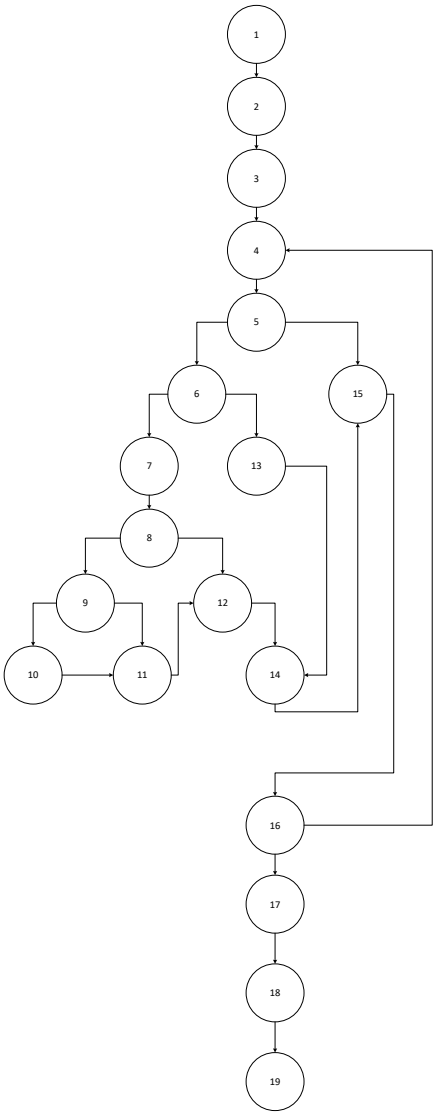
Path 2 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-20-21-23-24-25-26-27-28

Path 3 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-21-23-24-25-26-27-28

Path 4 : 1-2-3-4-5-6-7-8-9-22-23-24-25-26-27-28

Path 5 : 1-2-3-4-5-6-7-8-24-25-26-27-28

Maka gambar 4.1 disederhanakan berdasarkan kondisi atau simpul pada gambar 4.2.



Gambar 4.2 Hasil penyederhanaan *Flow Graph Streaming Webcam*

2. *Graph Matrix* Melakukan *Streaming Webcam*

Graph matrix dari algoritma melakukan *streaming webcam* dapat dilihat pada tabel 4.3.

Tabel 4.3 *Graph matrix* melakukan *streaming webcam*

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**SUM**
1		1																		0
2			1																	0
3				1																0
4					1															0
5						1									1					1
6							1						1							1
7								1												0
8									1			1								1
9										1	1									1
10											1									0
11												1								0
12														1						0
13														1						0
14															1					0
15																1				0
16																	1			0
17																		1		0
18																			1	0
Total																				4

V(G) = Jumlah *graph matrix* + 1

V(G) = 4 + 1

V(G) = 5

Berdasarkan pengujian yang dilakukan pada setiap metode, dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 5, Maka dapat disimpulkan bahwa pengujian *white box* pada proses pengujian melakukan *streaming webcam* berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.1.2 Pengujian Ekstrasi Ciri SIFT

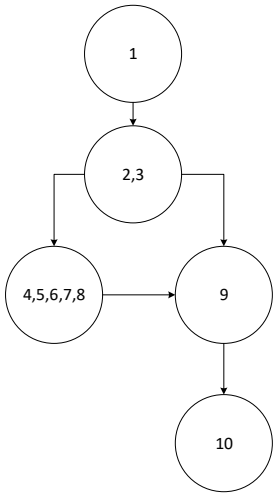
Pengujian ekstrasi ciri SIFT dapat dilihat pada tabel 4.4.

Tabel 4.4 Pengujian Ekstrasi Ciri SIFT

Line	Source
1	`private void ekstrasiCiriSIFT(Rect dataKonturTangan, Mat citraAsli) {`
2	`  Mat imgROI = citraAsli.submat(dataKonturTangan);`
3	`  if (imgROI != null) {`
4	`    sift.detectAndCompute(imgROI, new Mat(), kp, dp);`
5	`    Mat outputImage = new Mat(imgROI.rows(), imgROI.cols(), Imgcodecs.CV_LOAD_IMAGE_COLOR);`
6	`    Features2d.drawKeypoints(imgROI, kp, outputImage);`
7	`    Mat submat = citraAsli.submat(new Rect(dataKonturTangan.tl(), dataKonturTangan.br()));`
8	`    outputImage.copyTo(submat);`
9	`  }`
10	`}`

1. *Flow Graph* Ekstrasi Ciri SIFT

Berdasarkan tabel 4.4 maka dapat dibentuk *flow graph* seperti pada gambar 4.4.



Gambar 4.3 *Flow Graph* Ekstrasi Ciri SIFT

Dari gambar 4.4 dapat dihitung *cylomatic complexity* sebagai berikut :

V(G) = E – N + 2

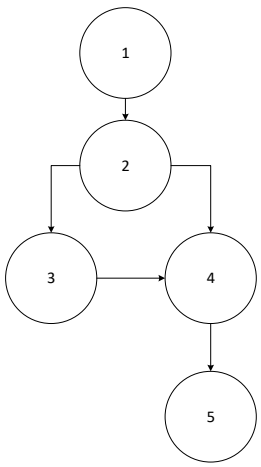
V(G) = 5 – 5 + 2 = 2

Jadi, *cyclomatic complexity* untuk gambar 4.4 adalah 2. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 1 *path* yang terdiri dari:

Path 1 : 1-2-3-4-5-6-7-8-9-10

Path : 1-2-3-9-10

Maka gambar 4.4 disederhanakan berdasarkan kondisi atau simpul pada gambar 4.5.



Gambar 4.4 Hasil penyederhanaan *Flow Graph* Ekstrasi Ciri SIFT

2. *Graph Matrix* Ekstrasi Ciri SIFT

Graph matrix dari algoritma ekstrasi ciri SIFT dapat dilihat pada tabel 4.5.

Tabel 4.5 Graph Matrix Ekstrasi Ciri SIFT

	1	2	3	4	5	SUM
1		1				0
2			1	1		1
3				1		0
4					1	0
Total						1

V(G) = Jumlah graph matrix + 1

V(G) = 1 + 1

V(G) = 2

Berdasarkan pengujian yang dilakukan pada setiap metode, dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 2, Maka dapat disimpulkan bahwa pengujian *white box* pada proses pengujian ekstrasi ciri SIFT berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.1.3 Pengujian Training KNN

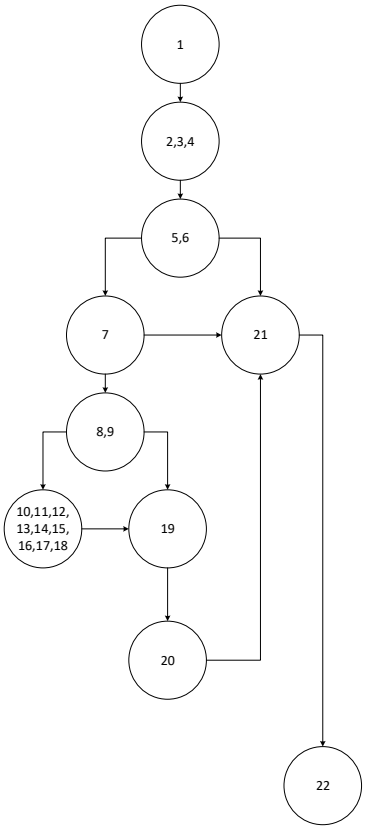
Pengujian training KNN dapat dilihat pada tabel 4.6.

Tabel 4.6 Pengujian Training KNN

Line	Source
1	`private void trainKNN() {`
2	`  int counts1 = 0;`
3	`  int counts2 = 0;`
4	`    for (String f : new File(path).list()) {`
5	`      counts1++;`
6	`      if (!f.equals("Thumbs.db")) {`
7	`       for (File data_ : new File("gambar/" + f + "/").listFiles()) {`
8	`        counts2++;`
9	`        if (!data_.getAbsolutePath().contains("Thumbs.db")) {`
10	`         Mat num = Imgcodecs.imread(data_.getAbsolutePath());`
11	`         sift.detectAndCompute(num, new Mat(), kp, dp);`
12	`         dp.convertTo(dp, CvType.CV_32F);`
13	`         Mat resizeimage = new Mat();`
14	`         Size sz = new Size(128, 85);`
15	`         Imgproc.resize(dp, resizeimage, sz);`
16	`         Mat labelsMat = new Mat(1, 1, CvType.CV_32SC1, new Scalar(counts1));`
17	`         trainData.push_back(resizeimage.reshape(1, 1));`
18	`         trainLabel.push_back(labelsMat);`
19	`        }`
20	`       }`
21	`      }`
22	`    }`

1. *Flow Graph* Training KNN

Berdasarkan tabel 4.6 maka dapat dibentuk *flow graph* seperti pada gambar 4.5.



Gambar 4.5 *Flow Graph* Training KNN

Dari gambar 4.5 dapat dihitung *cylomatic complexity* sebagai berikut :

V(G) = E – N + 2

V(G) = 12 – 10 + 2 = 4

Jadi, *cyclomatic complexity* untuk gambar 4.5 adalah 4. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 3 *path* yang terdiri dari:

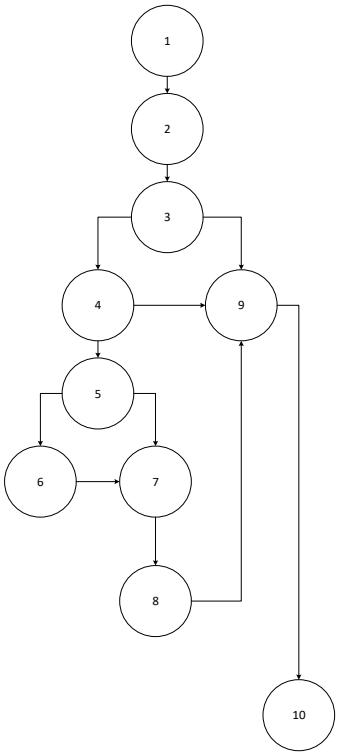
Path 1 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22

Path 2 : 1-2-3-4-5-6-7-8-9-19-20-21-22

Path 3 : 1-2-3-4-5-6-7-21-22

Path 4 : 1-2-3-4-5-6-21-22

Maka gambar 4.5 disederhanakan berdasarkan kondisi atau simpul pada gambar 4.6.



Gambar 4.6 Hasil penyederhanaan *training KNN*

2. *Graph Matrix* training KNN

Graph matrix dari algoritma training KNN dapat dilihat pada tabel 4.7.

Tabel 4.7 Graph Matrix Training KNN

	1	2	3	4	5	6	7	8	9	10	SUM
1		1									0
2			1								0
3				1					1		1
4					1			1			1
5						1	1				1
6							1				0
7								1			0
8									1		0
9										1	0
Total											3

V(G) = Jumlah graph matrix + 1

V(G) = 3 + 1

V(G) = 4

Berdasarkan pengujian yang dilakukan pada setiap metode, dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 4, Maka dapat disimpulkan bahwa pengujian *white box* pada proses pengujian training KNN berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.1.4 Pengujian Testing KNN

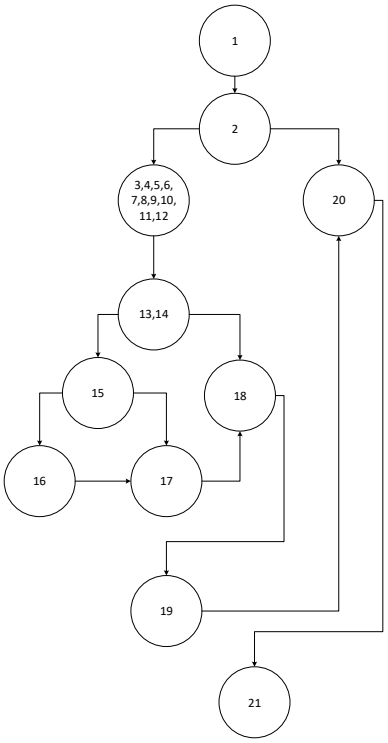
Pengujian testing KNN dapat dilihat pada tabel 4.8.

Tabel 4.8 Pengujian Testing KNN

Line	Source
1	`private void testKNN(Mat img) {`
2	`if (knn != null) {`
3	`sift.detectAndCompute(img, new Mat(), kp, dp);`
4	`dp.convertTo(dp, CvType.CV_32F);`
5	`Mat resizeimage2 = new Mat();`
6	`Size sz2 = new Size(128, 85);`
7	`Imgproc.resize(dp, resizeimage2, sz2);`
8	`resizeimage2.convertTo(resizeimage2, CvType.CV_32F);`
9	`Mat res = new Mat();`
10	`float hasil = knn.findNearest(resizeimage2.reshape(1, 1), 3, res);`
11	`int counts = 0;`
12	`for (String f : new File(pathresize).list()) {`
13	`counts++;`
14	`if (!f.equals("Thumbs.db")) {`
15	`if (hasil == counts) {`
16	`label = f;`
17	`}`
18	`}`
19	`}`
20	`}`
21	`}`

1. *Flow Graph* Testing KNN

Berdasarkan tabel 4.8 maka dapat dibentuk *flow graph* seperti pada gambar 4.7.



Gambar 4.7 *Flow Graph* Testing KNN

Dari gambar 4.7 dapat dihitung *cylomatic complexity* sebagai berikut :

V(G) = E – N + 2

V(G) = 13 – 11 + 2 = 4

Jadi, *cyclomatic complexity* untuk gambar 4.7 adalah 4. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 4 *path* yang terdiri dari:

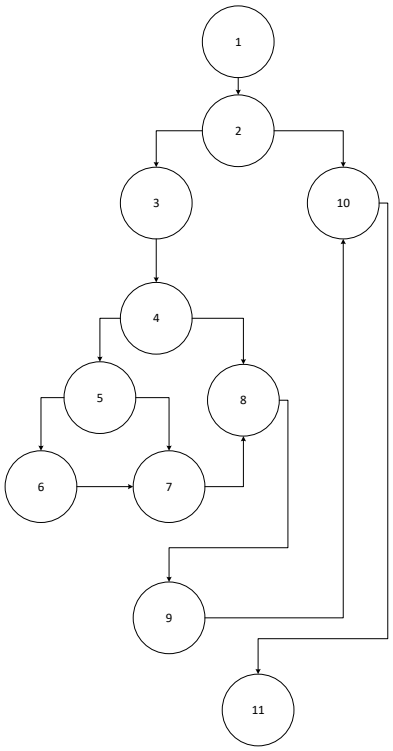
Path 1 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21

Path 2 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-17-18-19-20-21

Path 3 : 1-2-3-4-5-6-7-8-9-10-11-12-13-14-18-19-20-21

Path 4 : 1-2-20-21

Maka gambar 4.7 disederhanakan berdasarkan kondisi atau simpul pada gambar 4.8.



Gambar 4.8 Hasil penyederhanaan *testing KNN*

2. *Graph Matrix* testing KNN

Graph matrix dari algoritma testing KNN dapat dilihat pada tabel 4.9.

Tabel 4.9 Graph Matrix Testing KNN

	1	2	3	4	5	6	7	8	9	10	11	SUM
1		1										0
2			1							1		1
3				1								0
4					1			1				1
5						1	1					1
6							1					0
7								1				0
8									1			0
9										1		0
10											1	0
Total												3

V(G) = Jumlah graph matrix + 1

V(G) = 3 + 1

V(G) = 4

Berdasarkan pengujian yang dilakukan pada setiap metode, dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 4, Maka dapat disimpulkan bahwa pengujian *white box* pada proses pengujian testing KNN berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.2 Pengujian Akurasi

Pengujian akurasi pada penelitian ini dilakukan untuk mengetahui nilai akurasi, dalam implementasinya pengujian performasi menggunakan metode *confusion matrix*. *Confusion matrix* adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi pada konsep data mining atau Sistem Pendukung Keputusan. Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN). Nilai *True Negative* (TN) merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan *False Positive* (FP) merupakan data negatif namun terdeteksi sebagai data positif. Sementara itu, *True Positive* (TP) merupakan data positif yang terdeteksi benar. *False Negative* (FN) merupakan kebalikan dari *True Positive*, sehingga data posifit, namun terdeteksi sebagai data negatif[18].

Pengujian ini dilakukan untuk menguji stabilitas akurasi jika diuji dengan data latih dan data uji yang berbeda. Pada pengujian ini, data yang digunakan sebanyak 105 data yang dibagi menjadi 7 bagian untuk menguji akurasi dari setiap kelas, dimana pada setiap bagian diuji sebanyak 15 data.

4.2.2.1 Pengujian akurasi ke-1

Pengujian pertama dilakukan pada kelas Play. Berikut ini merupakan hasil pengujian dari kelas Play pada Tabel 4.10 sebagai berikut :

Tabel 4.10 Pengujian akurasi ke-1

No	Frame	Kelas Asli	Hasil Klasifikasi
1	Play-1	Play	Play
2	Play-2	Play	Play
3	Play-3	Play	Play
4	Play-4	Play	Play
5	Play-5	Play	Play
6	Play-6	Play	Pause
7	Play-7	Play	Pause
8	Play-8	Play	Play
9	Play-9	Play	Play
10	Play-10	Play	Volume Up
11	Play-11	Play	Pause
12	Play-12	Play	Play
13	Play-13	Play	Pause
14	Play-14	Play	Pause
15	Play-15	Play	Pause

Berdasarkan hasil pengujian yang dilakukan pada kelas Play didapatkan hasil, yaitu 8 buah data benar dan sisanya 7 buah data salah pada hasil prediksinya.

4.2.2.2 Pengujian akurasi ke-2

Pengujian kedua dilakukan pada kelas Pause. Berikut ini merupakan hasil pengujian dari kelas Pause pada Tabel 4.11 sebagai berikut :

Tabel 4.11 Pengujian akurasi ke-2

No	Citra	Kelas Asli	Hasil Klasifikasi
1	Pause-1	Pause	Pause
2	Pause-2	Pause	Pause
3	Pause-3	Pause	Pause
4	Pause-4	Pause	Pause
5	Pause-5	Pause	Pause
6	Pause-6	Pause	Pause

7	Pause-7	Pause	Play
8	Pause-8	Pause	Pause
9	Pause-9	Pause	Pause
10	Pause-10	Pause	Pause
11	Pause-11	Pause	Pause
12	Pause-12	Pause	Pause
13	Pause-13	Pause	Play
14	Pause-14	Pause	Play
15	Pause-15	Pause	Pause

Berdasarkan hasil pengujian yang dilakukan pada kelas Pause didapatkan hasil, yaitu 12 buah data benar dan sisanya 3 buah data salah pada hasil prediksinya.

4.2.2.3 Pengujian akurasi ke-3

Pengujian pertama dilakukan pada kelas Stop. Berikut ini merupakan hasil pengujian dari kelas Stop pada Tabel 4.12 sebagai berikut :

Tabel 4.12 Pengujian akurasi ke-3

No	Citra	Kelas Asli	Hasil Klasifikasi
1	Stop-1	Stop	Volume Up
2	Stop-2	Stop	Volume Up
3	Stop-3	Stop	Stop
4	Stop-4	Stop	Stop
5	Stop-5	Stop	Stop
6	Stop-6	Stop	Stop
7	Stop-7	Stop	Stop
8	Stop-8	Stop	Stop
9	Stop-9	Stop	Stop
10	Stop-10	Stop	Stop
11	Stop-11	Stop	Stop
12	Stop-12	Stop	Stop
13	Stop-13	Stop	Stop
14	Stop-14	Stop	Stop
15	Stop-15	Stop	Volume Up

Berdasarkan hasil pengujian yang dilakukan pada kelas Stop didapatkan hasil, yaitu 12 buah data benar dan sisanya 3 buah data salah pada hasil prediksinya.

4.2.2.4 Pengujian akurasi ke-4

Pengujian pertama dilakukan pada kelas Previous. Berikut ini merupakan hasil pengujian dari kelas Previous pada Tabel 4.13 sebagai berikut :

Tabel 4.13 Pengujian akurasi ke-4

No	Citra	Kelas Asli	Hasil Klasifikasi
1	Previous-1	Previous	Previous
2	Previous-2	Previous	Previous
3	Previous-3	Previous	Stop
4	Previous-4	Previous	Stop
5	Previous-5	Previous	Stop
6	Previous-6	Previous	Stop
7	Previous-7	Previous	Previous
8	Previous-8	Previous	Previous
9	Previous-9	Previous	Previous
10	Previous-10	Previous	Stop
11	Previous-11	Previous	Stop
12	Previous-12	Previous	Stop
13	Previous-13	Previous	Stop
14	Previous-14	Previous	Stop
15	Previous-15	Previous	Stop

Berdasarkan hasil pengujian yang dilakukan pada kelas Previous didapatkan hasil, yaitu 5 buah data benar dan sisanya 10 buah data salah pada hasil prediksinya.

4.2.2.5 Pengujian akurasi ke-5

Pengujian pertama dilakukan pada kelas Next. Berikut ini merupakan hasil pengujian dari kelas Next pada Tabel 4.14 sebagai berikut :

Tabel 4.14 Pengujian akurasi ke-5

No	Citra	Kelas Asli	Hasil Klasifikasi
1	Next-1	Next	Stop
2	Next-2	Next	Stop
3	Next-3	Next	Stop
4	Next-4	Next	Next
5	Next-5	Next	Stop
6	Next-6	Next	Stop

7	Next-7	Next	Stop
8	Next-8	Next	Stop
9	Next-9	Next	Next
10	Next-10	Next	Stop
11	Next-11	Next	Stop
12	Next-12	Next	Stop
13	Next-13	Next	Next
14	Next-14	Next	Stop
15	Next-15	Next	Next

Berdasarkan hasil pengujian yang dilakukan pada kelas Next didapatkan hasil, yaitu 4 buah data benar dan sisanya 11 buah data salah pada hasil prediksinya.

4.2.2.6 Pengujian akurasi ke-6

Pengujian pertama dilakukan pada kelas Volume Up. Berikut ini merupakan hasil pengujian dari kelas Volume Up pada Tabel 4.15 sebagai berikut :

Tabel 4.15 Pengujian akurasi ke-6

No	Citra	Kelas Asli	Hasil Klasifikasi
1	Volume Up-1	Volume Up	Stop
2	Volume Up-2	Volume Up	Stop
3	Volume Up-3	Volume Up	Stop
4	Volume Up-4	Volume Up	Stop
5	Volume Up-5	Volume Up	Volume Up
6	Volume Up-6	Volume Up	Play
7	Volume Up-7	Volume Up	Volume Up
8	Volume Up-8	Volume Up	Stop
9	Volume Up-9	Volume Up	Volume Up
10	Volume Up-10	Volume Up	Stop
11	Volume Up-11	Volume Up	Stop
12	Volume Up-12	Volume Up	Stop
13	Volume Up-13	Volume Up	Volume Up
14	Volume Up-14	Volume Up	Play
15	Volume Up-15	Volume Up	Volume Up

Berdasarkan hasil pengujian yang dilakukan pada kelas Volume Up didapatkan hasil, yaitu 5 buah data benar dan sisanya 10 buah data salah pada hasil prediksinya.

4.2.2.7 Pengujian akurasi pada 7-fold

Pengujian pertama dilakukan pada kelas Volume Down. Berikut ini merupakan hasil pengujian dari kelas Volume Down pada Tabel 4.16 sebagai berikut :

Tabel 4.16 Pengujian akurasi pada 7-fold

No	Citra	Kelas Asli	Hasil Klasifikasi
1	Volume Down-1	Volume Down	Pause
2	Volume Down-2	Volume Down	Volume Down
3	Volume Down-3	Volume Down	Volume Down
4	Volume Down-4	Volume Down	Pause
5	Volume Down-5	Volume Down	Volume Down
6	Volume Down-6	Volume Down	Pause
7	Volume Down-7	Volume Down	Volume Down
8	Volume Down-8	Volume Down	Pause
9	Volume Down-9	Volume Down	Volume Down
10	Volume Down-10	Volume Down	Pause
11	Volume Down-11	Volume Down	Pause
12	Volume Down-12	Volume Down	Pause
13	Volume Down-13	Volume Down	Pause
14	Volume Down-14	Volume Down	Pause
15	Volume Down-15	Volume Down	Pause

Berdasarkan hasil pengujian yang dilakukan pada kelas Volume Down didapatkan hasil, yaitu 5 buah data benar dan sisanya 10 buah data salah pada hasil prediksinya.

4.2.3 Hasil Pengujian

Setelah dilakukan perhitungan pengujian akurasi pada setiap kelas, langkah selanjutnya adalah menghitung keseluruhan akurasi menggunakan metode *confusion matrix*. Langkah pertama adalah ubah semua data pengujian akurasi ke dalam *confusion matrix* seperti pada table 4.18 dibawah berikut ini :

Tabel 4.17 *Confusion Matrix*

		Kelas Hasil Prediksi						
		Play	**Pause**	**Stop**	**Previous**	**Next**	**Volume Up**	**Volume Down**
Kelas Asli	**Play**	8	6	0	0	0	1	0
	Pause	3	12	0	0	0	0	0
	Stop	0	0	12	0	0	3	0
	Previous	0	0	10	5	0	0	0
	Next	0	0	12	0	4	0	0
	Volume Up	2	0	8	0	0	5	0
	Volume Down	0	10	0	0	0	0	5

Kemudian langkah selanjutnya adalah menghitung nilai presisi, *recall*, akurasi menggunakan rumus berikut ini :

$$Presisi = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Akurasi = \frac{TP}{TP + TN + FP + FN}$$

Pertama hitung nilai presisi terlebih dahulu pada setiap kelas. Perhitungan presisi pada setiap kelas dapat dilihat dibawah ini :

$Presisi\ Play = \frac{8}{8+3+0+0+0+2+0}$ = 61.53%

$Presisi\ Pause = \frac{12}{12+6+0+0+0+0+10}$ = 42.85%

$Presisi\ Stop = \frac{12}{12+0+0+10+2+8+0}$ = 37.5%

$Presisi\ Previous = \frac{5}{5+0+0+0+0+0+0}$ = 100%

$Presisi\ Next = \frac{4}{4+0+0+0+0+0+0}$ = 100%

$Presisi\ Volume\ Up = \frac{5}{5+1+0+3+0+0+0}$ = 55.55%

$Presisi\ Volume\ Down = \frac{5}{5+0+0+0+0+0+0}$ = 100%

Kemudian kita hitung nilai rata-rata presisi keseluruhan kelas, maka hasilnya sebagai berikut :

$Presisi = \frac{Play+Pause+Stop+Previous+Next+Volume\ Up+Volume\ Down}{Total\ jumlah\ kelas}$

$Presisi = \frac{61.53+42.85+37.5+100+100+55.55+100}{7}$ =71.06%

Kedua hitung nilai *recall* pada setiap kelas. Perhitungan *recall* pada setiap kelas dapat dilihat dibawah ini :

$Recall\ Play = \frac{8}{8+6+0+0+0+1+0}$ = 53.33%

$Recall\ Pause = \frac{12}{12+3+0+0+0+0+0}$ = 80%

$Recall\ Stop = \frac{12}{12+0+0+0+0+3+0}$ = 80%

$Recall\ Previous = \frac{5}{5+0+0+11+0+0+0}$ = 33.33%

$Recall\ Next = \frac{4}{4+0+0+12+0+0+0}$ = 26.66%

$Recall\ Volume\ Up = \frac{5}{5+2+0+8+0+0+0}$ = 33.33%

$Recall\ Volume\ Down = \frac{5}{6+0+10+0+0+0+0}$ = 33.33%

Kemudian kita hitung nilai rata-rata *recall* keseluruhan kelas, maka hasilnya sebagai berikut :

$$Recall = \frac{Play+Pause+Stop+Previous+Next+Volume\ Up+Volume\ Down}{Total\ jumlah\ kelas}$$

$$Recall = \frac{53.33+80+80+33.33+26.66+33.33+33.33}{7} = 48.56\%$$

Langkah terakhir adalah menghitung total akurasi dari *confusion matrix*, perhitungannya dibawah ini :

$$Akurasi = \frac{8+12+12+5+4+5+5}{105} = 48.56\%$$

Dari data tersebut dapat diambil kesimpulan bahwa hasil pengujian menggunakan *confusion matrix* didapatkan akurasi sebesar 48.56%. Hal ini disebabkan karena fitur vektor pada SIFT terjadi pengurangan informasi karena faktor *resize image* untuk mendapatkan masukkan antar data *training* memiliki dimensi yang sama.