BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

Pada bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi yang dibangun. Tahapan ini dilakukan setelah analisis dan perancangan selesai dilakukan dan selanjutnya akan diimplementasikan kedalam bahasa pemrograman. Setelah implementasi maka dilakukan pengujian terhadap aplikasi. Aplikasi yang telah dibangun akan diuji untuk mengetahui apakah sistem tersebut dapat berjalan sesuai dengan tujuannya atau tidak

4.1. Implementasi

Tahap implementasi adalah proses pembangunan perangkat lunak, tahap kelanjutan dari kegiatan perancangan sistem yang bertujuan untuk mengkonfirmasi perancangan sistem yang telah dibuat pada bab sebelumnya.

4.1.1. Implementasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam pembangunan sistem ini dijelaskan pada tabel Tabel 4.1.

Tabel 4.1 Perangkat Keras Yang Digunakan

Tipe Perangkat	Spesifikasi Perangkat
Prosesor	Intel Core I7-3610QM 2.3Ghz
Ram	6 GB *DDR3 Dual Channel*
VGA	*Nvidia GT 630M 2GB*
Layar	LCD 14 “
Penyimpanan	256 GB *SSD (Solid State Disk)*
Keyboard	*Laptop Keyboard*
Mouse	*Logitech Wireless Mouse M185*
Kamera	*Logitech C170 5 MP*
Papan Prototipe	*Arduino Uno Rev 3, Bread board (*840 lubang*)*

Servo	1. 1 x *Hitech HS-805bb+ (24.7Kg/Cm)* 2. 2 x *GWS(Grand Wing System)S03N/STD/J (3.2Kg/Cm)* 3. 2 x *GWS(Grand Wing System)S03T/STD/F (8Kg/Cm)*
Power Supply	1. *Montana Regulated Power Supply 5-13.8V 7A*
Perangkat lain	1. Rangka alat bantu (*Alumunium*) 2. *Rainbow Flat Cable 2 M* 3. Beban Pemberat (*2 x Stainless Steel)* 4. *Usb Extension 1.5 M*

4.1.2. Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pembangunan sistem ini dijelaskan pada tabel Tabel 4.2.

Tabel 4.2 Perangkat Lunak Yang Digunakan

Tipe Perangkat	Spesifikasi Perangkat
Sistem Operasi	*Windows 7 Ultimate SP1 64 Bit*
Bahasa Pemprograman	*C#*
Compiler	*Microsoft Visual Studio Ultimate 2012, Arduino 1.63*
FrameWork	*Emgu CV 2.4.10.1940*

4.1.3. Implementasi Antarmuka

Pada tahap ini dilakukan penerapan hasil perancangan antarmuka kedalam sistem yang dibangun dengan menttunakan perangkat lunak yang telah dipaparkan pada sub bab implementasi perangkat lunak. Berikut ini beberapa tampilan antarmuka yang telah diimplementasikan.

4.1.4. Implementasi Antarmuka Tampilan Menu Utama

Implementasi antarmuka tampilan menu utama digunakan untuk mengatur pilihan kamera port, pilihan port *Arduino*, kalibrasi papan permainan dan pencarian kepemilikan dari suatu petak properti. Implementasi antarmuka tampilan menu utama dapat dilihat pada Lampiran A1.

4.1.5. Implementasi Antarmuka Tampilan Menu Pengaturan Petak dan Lokasi Papan Permainan

Implementasi antar muka tampilan menu pengaturan petak dan lokasi papan permainan digunakan untuk mengatur petak dan lokasi papan permainan, mengatur kecepatan dan test koordinat. Implementasi antar muka tampilan menu pengaturan petak dan lokasi papan permainan dapat dilihat pada Lampiran A2.

4.1.6. Implementasi Antarmuka Tampilan Menu Pengaturan Pemain

Implementasi antarmuka tampilan menu pengaturan pemain digunakan untuk mengatur pemain. Implementasi antarmuka tampilan menu pengaturan pemain dapat dilihat dengan Lampiran A3.

4.2. Pengujian Sistem

Pengujian sistem dilakukan bertujuan untuk menemukan kesalahan atau kekurangan pada perangkat lunak yang telah diuji. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut. Dalam penelitian ini pengujian yang dilakukan terhadap sistem yaitu pengujian secara fungsional (*Alpha*). Metode yang digunakan dalam pengujian ini adalah pengujian *Black Box* yang berfokus pada persyaratan fungsional dari sistem yang dibangun dan *whitebox* yang berfokus pada alur aplikasi yang dibangun.

4.2.1. Pegujian Alpha

Pengujian *Alpha* dilakukan pada sisi pengembangan yang merekam semua kesalahan dan masalah pemakaian. Pengujian *Alpha* dilakukan pada sebuah lingkungan yang terkendali.

4.2.2. Skenario Pengujian Alpha

Rencana pengujian adalah pengujian terhadap fungsionalitas yang terdapat dalam aplikasi, apakah fungsionitas tersebut sesuai yang diharapkan atau tidak. Rencana pengujian dari sitem yang dibangun dapat dilihat pada Tabel 4.3.

Tabel 4.3 Skenario Pengujian *Alpha*

No	Komponen yang Diuji	Skenario Pengujian	Jenis Pengujian
1	Fungsionalitas Mengatur Petak	Menambah Petak	*Black Box*
		Mengubah Petak	*Black Box*
		Menghapus Petak	*Black Box*
2	Fungsionalitas Mengatur Pemain	Menambah Pemain	*Black Box*
		Mengubah Pemain	*Black Box*
		Menghapus Pemain	*Black Box*
3	Fungsionalias Mengatur Lokasi Penyimpanan Gambar Sampel	Memilih Lokasi Penyimpanan Gambar Sampel	*Black Box*
4	Fungsionalitas Mengatur Kamera dan Port *Arduino*	Memilih Kamera	*Black Box*
		Memilih Port *Arduino*	*Black Box*
5	Fungsionalitas Kecepatan dan Test Koordinat Alat Bantu	Mengatur Test Koordinat Alat Bantu	*Black Box*
		Mengatur Kecepatan Alat Bantu	*Black Box*
6	Deteksi Petak	Mendeteksi Petak Permainan	*WhiteBox*

4.2.3. Pengujian Black Box

Pada tahap pengujian berisi pemaparan dari rencana pengujian yang telah disusun pada skenario pengujian. Pengujian ini dilakukan menggunakan *Black Box* dengan hanya memperhatikan masukan ke dalam sistem dan keluaran dari masukan tersebut. Berdasarkan rencana pengujian, maka dapat dilakukan pengujian *Black Box* pada aplikasi sebagai berikut :

4.2.3.1. Pengujian Mengatur Petak

Pengujian Mengatur Petak digunakan untuk mengatur data petak yang akan digunakan sebagai acuan koordinat alat bantu, kalibrasi papan permainan dan deteksi kepemilikan properti. Pengujian ini dibagi menjadi beberapa butir pengujian yaitu menambahkan data petak dan mengubah data petak. Pengujian menambahkan data petak yang benar dapat dilihat pada Tabel 4.4. Pada pengujian menambahkan data petak, *user* memilih tipe pengaturan menjadi atur untuk menambahkan petak tertentu. Pengujian menambahkan data petak diuji dari paramter masukan seperti nama, lokasi dan koordinat petak. Pengisian lokasi petak dan koordinat petak menggunakan *numeric up down* yang apabila diisi dengan nilai kosong ataupun negatif, maka secara *default* akan otomatis kembali menjadi angka 1 untuk lokasi petak dan angka 0 untuk koordinat petak, yang kemudian memvalidasi apakah ada lokasi yang sama. Pada lokasi petak, digunakan untuk membedakan petak berdasarkan posisi petak seperti papan monopoli asli yang memiliki jumlah petak tiap sisinya 10 dan terdapat 4 sisi. Pada koordinat petak, koordinat ini berguna untuk menggerakan alat bantu *servo* atau motor penggerak ke derajat tertentu. Pengujian mengubah data petak dilihat pada Tabel 4.5. Apabila *user* memilih data petak dan kemudian menekan tombol ubah, secara automatis tombol tambah akan berubah menjadi tombol selesai untuk mengakhiri proses pengubahan data petak. Pengujian menghapus data petak dapat dilihat pada Tabel 4.6.

Tabel 4.4 Pengujian Menambahkan data Petak

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* memilih tipe pengaturan = Atur 2. *User* mengisi data	Data Petak Tersimpan dan menampilkan pemberitahuan	Sistem menyimpan data petak dan menampilkan pesan	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	Nama = “Petak” Lokasi = “1,1” Koordinat = “10,45,23,11,42” 3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol tambah		"Data Berhasil Ditambah"	
2	1.*User* memilih tipe pengaturan = Atur 2.*User* mengisi data Nama=” ” Lokasi = “ 2,1“ Koordinat = “10,45,23,11,42 “ 3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol tambah	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan kesalahan “Nama Tidak Boleh Kosong”	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
3	1.*User* memilih tipe pengaturan = Atur 2.*User* mengisi data Nama=”Monopoli” Lokasi = “ , “ (Asumsi Lokasi tidak ada yang bernilai “1,1”) Koordinat = “10,45,23,11,42 “ 3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol tambah	Data Petak Tersimpan dan menampilkan pemberitahuan	Sistem menyimpan data petak dan menampilkan pesan "Data Berhasil Ditambah"	Sesuai
4	1.*User* memilih tipe pengaturan = Atur 2.*User* mengisi data Nama=”Monopoli” Lokasi = “ , “ (Asumsi Lokasi ada yang bernilai “1,1”) Koordinat = “10,45,23,11,42 “	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan kesalahan “Terdapat Petak dengan lokasi yang sama”	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol tambah			
5	1.*User* memilih tipe pengaturan = Atur 2.*User* mengisi data Nama=”Monopoli” Lokasi = “1,2“ Koordinat = “ , , , , “ (Asumsi Koordinat tidak ada yang bernilai “0,0,0,0,0”) 3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol tambah	Data Petak Tersimpan dan menampilkan pemberitahuan	Sistem menyimpan data petak dan menampilkan pesan "Data Berhasil Ditambah"	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
6	1.*User* memilih tipe pengaturan = Atur 2.*User* mengisi data Nama=”Monopoli” Lokasi = “1,2“ Koordinat = “ , , , , “ (Asumsi Koordinat ada yang bernilai “0,0,0,0,0”) 3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol tambah	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan kesalahan “Terdapat Petak dengan lokasi yang sama”	Sesuai
7	1.*User* memilih tipe pengaturan = Atur 2.*User* mengisi data Nama=”Monopoli” Lokasi = “1,2“ Koordinat = “10,45,23,11,42“ (Asumsi *user* melewati	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan kesalahan “Gambar belum dipilih”	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	penambahan data gambar) 4.*User* menekan tombol tambah			
8	1.*User* memilih tipe pengaturan = Atur 2.*User* mengisi data Nama=”Monopoli” Lokasi = “1,2“ Koordinat = “10,45,23,11,42“ 3.*User* menekan cari gambar 4.*User* menekan cancel (Asumsi *user* membatalkan penambahan data gambar) 5.*User* menekan tombol tambah	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan kesalahan “Gambar belum dipilih”	Sesuai

Tabel 4.5 Pengujian Mengubah data Petak

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* memilih data petak (Asumsi data petak masih kosong)	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	2.*User* menekan tombol ubah		kesalahan “Data Petak Kosong”	
2	1.*User* memilih data petak (Asumsi data petak tidak kosong) 2.*User* menekan tombol ubah 3.*User* mengisikan data Nama: “Marble” Lokasi: “4,3” Koordinat:”12,45,24,11,42” 3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol Selesai	Data Petak Diubah dan menampilkan pemberitahuan	Sistem mengubah data petak dan menampilkan pesan "Data Berhasil Diubah"	Sesuai
3	1.*User* memilih data petak (Asumsi data petak tidak kosong) 2.*User* menekan tombol ubah 3.*User* mengisikan data Nama: “ ” Lokasi:”4,3” Koordinat:”12,45,24,11,42”	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan "Nama tidak boleh kosong"	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol Selesai			
4	1.*User* memilih data petak (Asumsi data petak tidak kosong) 2.*User* menekan tombol ubah 3.*User* mengisikan data Nama: “Marble” Lokasi:” , ” (Asumsi lokasi tidak ada yang bernilai “1,1”) Koordinat:”12,45,24,11,42” 3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol Selesai	Data Petak Diubah dan menampilkan pemberitahuan	Sistem mengubah data petak dan menampilkan pesan "Data Berhasil Diubah"	Sesuai
5	1.*User* memilih data petak (Asumsi data petak tidak kosong)	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	2.*User* menekan tombol ubah 3.*User* mengisikan data Nama: “Marble” Lokasi:” , ” (Asumsi lokasi ada yang bernilai “1,1”) Koordinat:”12,45,24,11,42” 3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol Selesai		"Terdapat petak dengan lokasi yang sama"	
6	1.*User* memilih data petak (Asumsi data petak tidak kosong) 2.*User* menekan tombol ubah 3.*User* mengisikan data Nama: “Marble” Lokasi:”1,2” Koordinat:” , , , , ” (Asumsi koordinat tidak ada yang bernilai “0,0,0,0,0”)	Data Petak Diubah dan menampilkan pemberitahuan	Sistem mengubah data petak dan menampilkan pesan "Data Berhasil Diubah"	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol Selesai			
7	1.*User* memilih data petak (Asumsi data petak tidak kosong) 2.*User* menekan tombol ubah 3.*User* mengisikan data Nama: “Marble” Lokasi:”1,2” Koordinat:” , , , , ” (Asumsi koordinat ada yang bernilai “0,0,0,0,0”) 3.*User* menekan cari gambar 4.*User* memilih data gambar 5.*User* menekan Open 6.*User* menekan tombol Selesai	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan "Terdapat petak dengan lokasi yang sama"	Sesuai
8	1.*User* memilih data petak (Asumsi data petak tidak kosong)	Data Petak Diubah dan	Sistem mengubah data petak	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	2.*User* menekan tombol ubah 3.*User* mengisikan data Nama: “Marble” Lokasi:”1,2” Koordinat:”12,45,24,11,42” (Asumsi *user* tidak mengubah gambar) 4.*User* menekan tombol Selesai	menampilkan pemberitahuan	dan menampilkan pesan "Data Berhasil Diubah"	
9	1.*User* memilih data petak (Asumsi data petak tidak kosong) 2.*User* menekan tombol ubah 3.*User* mengisikan data (Asumsi *user* tidak mengubah nama petak) Lokasi:”1,2” Koordinat:”12,45,24,11,42” 4.*User* menekan cari gambar 5.*User* memilih data gambar 6.*User* menekan Open 7.*User* menekan tombol Selesai	Data Petak Diubah dan menampilkan pemberitahuan	Sistem mengubah data petak dan menampilkan pesan "Data Berhasil Diubah"	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
10	1.*User* memilih data petak (Asumsi data petak tidak kosong) 2.*User* menekan tombol ubah 3.*User* mengisikan data Nama: “Marble” (Asumsi *user* tidak mengubah lokasi petak) Koordinat:”12,45,24,11,42” 4.*User* menekan cari gambar 5.*User* memilih data gambar 6.*User* menekan Open 7.*User* menekan tombol Selesai	Data Petak Diubah dan menampilkan pemberitahuan	Sistem mengubah data petak dan menampilkan pesan "Data Berhasil Diubah"	Sesuai
11	1.*User* memilih data petak (Asumsi data petak tidak kosong) 2.*User* menekan tombol ubah 3.*User* mengisikan data Nama: “Marble” Lokasi:”1,2” (Asumsi *user* tidak mengubah koordinat petak)	Data Petak Diubah dan menampilkan pemberitahuan	Sistem mengubah data petak dan menampilkan pesan "Data Berhasil Diubah"	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	4.*User* menekan cari gambar 5.*User* memilih data gambar 6.*User* menekan Open 7.*User* menekan tombol Selesai			

Tabel 4.6 Pengujian Menghapus Data Petak

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* memilih data petak (Asumsi data petak masih kosong)	Menampilkan pesan kesalahan	Sistem menampilkan Pemberitahuan "Data Petak Kosong"	Sesuai
2	1.*User* memilih data petak (Asumsi data petak tidak kosong) 2.*User* menekan tombol hapus	Menghapus data petak dan tampil pemberitahuan	Sistem menghapus data dan menampilkan "Data berhasil dihapus"	Sesuai

4.2.3.2. Pengujian Mengatur Pemain

Pengujian Mengatur digunakan untuk mengatur data Pemain yang akan digunakan sebagai acuan nama dari pemilik kartu pada deteksi kepemilikan properti. Pengujian ini dibagi menjadi beberapa butir pengujian yaitu menambahkan data pemain, mengubah data pemain, dan menghapus data pemain. Pengujian menambahkan data pemain dapat dilihat pada Tabel 4.7, sedangkan

pengujian mengubah data pemain dapat dilihat pada Tabel 4.8. Pengujian menghapus data pemain dapat dilihat pada Tabel 4.9. Pengisian posisi pemain menggunakan *numericUpDown* yang apabila tidak diisi atau bernilai negatif maka secara automatis akan kembali ke nilai *default* 1. Pengisian tipe pemain adalah untuk membedakan antara pemain dengan tipe komputer dengan pemain biasa. Pengujian ini akan menguji parameter masukan seperti nama, posisi dan tipe pemain.

Tabel 4.7 Pengujian Menambahkan data Pemain

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* mengisi data Nama : “Acep” Posisi: “1” Checkbox Tipe Pemain = Komputer terpilih 2.*User* menekan tombol tambah	Data Pemain Tersimpan dan menampilkan pemberitahuan	Sistem menyimpan data Pemain dan menampilkan pesan "Data Berhasil Ditambah"	Sesuai
2	1.*User* mengisi data Nama : “ ” Posisi: “1” Checkbox Tipe Pemain = Komputer terpilih 2.*User* menekan tombol tambah	Menampilkan pesan kesalahan	Sistem menampilkan pesan kesalahan ” Nama tidak boleh kosong”	Sesuai
3	1.*User* mengisi data Nama : “Acep” Posisi: “ ”	Data Pemain Tersimpan dan	Sistem menyimpan data Pemain dan menampilkan pesan "Data	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	(Asumsi posisi tidak ada yang bernilai "1") Checkbox Tipe Pemain = Komputer terpilih 2.*User* menekan tombol tambah	menampilkan pemberitahuan	Berhasil Ditambah"	
4	1.*User* mengisi data Nama : "Acep" Posisi: " " (Asumsi posisi ada yang bernilai "1") Checkbox Tipe Pemain = Komputer terpilih 2.*User* menekan tombol tambah	Menampilkan pesan kesalahan	Sistem menampilkan pesan kesalahan " Terdapat pemain dengan posisi yang sama"	Sesuai
5	1.*User* mengisi data Nama : "Acep" Posisi: "1,2" Checkbox Tipe Pemain = Komputer terpilih (Asumsi tidak ada pemain lain bertipe komputer)	Data Pemain Tersimpan dan menampilkan pemberitahuan	Sistem menyimpan data Pemain dan menampilkan pesan "Data Berhasil Ditambah"	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	2.*User* menekan tombol tambah			
6	1.*User* mengisi data Nama : “Acep” Posisi: “1,2” Checkbox Tipe Pemain = Komputer terpilih (Asumsi ada pemain lain bertipe komputer) 2.*User* menekan tombol tambah	Menampilkan pesan kesalahan	Sistem menampilkan pesan kesalahan ” Terdapat pemain dengan tipe yang sama”	Sesuai
7	1.*User* mengisi data Nama : “Acep” Posisi: “1,2” Checkbox Tipe Pemain = Komputer tidak terpilih (Asumsi tidak ada pemain lain bertipe komputer dan jumlah pemain sebelum penambahan adalah 3) 2.*User* menekan tombol tambah	Menampilkan pesan kesalahan	Sistem menampilkan pesan kesalahan ” Minimal terdapat 1 pemain dengan tipe komputer”	Sesuai

Tabel 4.8 Pengujian Mengubah data Pemain

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* memilih data pemain 2.*User* memilih ubah (Asumsi data pemain kosong)	Menampilkan pesan kesalahan	Sistem menampilkan pesan "Data Pemain Kosong"	Sesuai
2	1.*User* memilih data pemain 2.*User* memilih ubah (Asumsi data pemain tidak kosong) 3.*User* mengisi data Nama:"Surasep" Posisi: "2" Checkbox Tipe Pemain = Komputer terpilih 4.*User* memilih selesai	Data Pemain Diubah dan menampilkan pemberitahuan	Sistem mengubah data Pemain dan menampilkan pesan "Data Berhasil Diubah"	Sesuai
3	1.*User* memilih data pemain 2.*User* memilih ubah (Asumsi data pemain tidak kosong) 3.*User* mengisi data Nama:" " Posisi: "2"	Menampilkan pesan kesalahan	Sistem menampilkan pesan "Nama tidak boleh kosong"	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	Checkbox Tipe Pemain = Komputer terpilih 4.*User* memilih selesai			
4	1.*User* memilih data pemain 2.*User* memilih ubah (Asumsi data pemain tidak kosong) 3.*User* mengisi data Nama:"Surasep" Posisi: " " (Asumsi tidak ada posisi yang bernilai "1") Checkbox Tipe Pemain = Komputer terpilih 4.*User* memilih selesai	Data Pemain Diubah dan menampilkan pemberitahuan	Sistem mengubah data Pemain dan menampilkan pesan "Data Berhasil Diubah"	Sesuai
5	1.*User* memilih data pemain 2.*User* memilih ubah (Asumsi data pemain tidak kosong) 3.*User* mengisi data Nama:"Surasep"	Menampilkan pesan kesalahan	Sistem menampilkan pesan "Terdapat pemain dengan posisi yang sama"	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	Posisi: " " (Asumsi ada posisi yang bernilai "1") Checkbox Tipe Pemain = Komputer terpilih 4.*User* memilih selesai			
6	1.*User* memilih data pemain 2.*User* memilih ubah (Asumsi data pemain tidak kosong) 3.*User* mengisi data Nama:"Surasep" Posisi: "2" Checkbox Tipe Pemain = Komputer terpilih (Asumsi terdapat pemain dengan tipe komputer) 4.*User* memilih selesai	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan kesalahan "Terdapat pemain dengan tipe yang sama"	Sesuai
7	1.*User* memilih data pemain 2.*User* memilih ubah	Menampilkan Pesan Kesalahan	Sistem menampilkan pesan kesalahan "Minimal terdapat	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	(Asumsi data pemain tidak kosong) 3.*User* mengisi data Nama:"Surasep" Posisi: "2" Checkbox Tipe Pemain = Komputer tidak terpilih (Asumsi tidak terdapat pemain dengan tipe komputer dan jumlah pemain sebelum ditambahkan berjumlah 3) 4.*User* memilih selesai		1 pemain dengan tipe komputer"	

Tabel 4.9 Pengujian Menghapus data Pemain

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* memilih data pemain (asumsi data pemain kosong)	Menampilkan pesan kesalahan	Sistem menampilkan pesan kesalahan "Data Pemain Kosong"	Sesuai
2	1.*User* memilih data pemain	Menghapus data dan	Sistem menghapus data dan	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	(Asumsi data pemain tidak kosong) 2.*User* memilih hapus	menampilkan pemberitahuan	menampilkan pemberitahuan "Data berhasil dihapus"	

4.2.3.3. Pengujian Kecepatan dan Test Koordinat Alat Bantu

Pengujian mengatur test koordinat alat bantu bertujuan untuk mengatur kecepatan dan mencoba koordinat dari alat bantu yang digunakan. Pengisian Test koordinat 1 samapai 5 adalah pengisian koordinat untuk menggerakan *servo* atau motor penggerak pada alat bantu. Koordinat ini akan menggerakan alat bantu pada *servo* tertentu berapa derajat harus bergerak. Pengisian kecepatan alat bantu secara *default* akan terpilih pada kecepatan manual dengan nilai *delay* atau nilai tunggu antar gerakan tiap satu derajat ke derajat tujuan adalah 65 ms. Pengisian kecepatan alat bantu menggunakan *numericUpDown*, apabila tidak diisi atau bernilai negatif makan secara *default* akan otomatis mengubah menjadi 1. Pengujian mencoba test koordinat dapat dilihat pada Tabel 4.10, sedangkan pengujian mengatur kecepatan dapat dilihat pada Tabel 4.11.

Tabel 4.10 Pengujian Memilih test koordinat Alat Bantu

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* mengisi data Test Koordinat 1 : "12" Test Koordinat 2 : "16"	Alat Bantu Bergerak	Sistem menggerakan alat bantu	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
	Test Koordinat 3 : "18" Test Koordinat 4 : "12" Test Koordinat 5 : "19" 2.*User* menekan coba koordinat			
1	1.*User* mengisi data Test Koordinat 1 : " " Test Koordinat 2 : " " Test Koordinat 3 : " " Test Koordinat 4 : " " Test Koordinat 5 : " " 2.*User* menekan coba koordinat	Alat Bantu Bergerak	Sistem menggerakan alat bantu	Sesuai

Tabel 4.11 Pengujian Mengatur Kecepatan

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	(Asumsi aplikasi baru berjalan dan *User* tidak mengatur kecepatan alat bantu)	Menetapkan kecepatan *default*	Sistem men-*check* kecepatan *default*	Sesuai
2	1.*User* memilih kecepatan selain manual	Menetapkan pilihan kecepatan	Sistem men-*check* kecepatan pilihan *user*	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
3	1.*User* memilih kecepan manual 2.*User* mengisi data manual Delay:"20"	Menetapkan pilihan kecepatan	Sistem men-*check* kecepatan pilihan *user*	Sesuai
4	1.*User* memilih kecepan manual 2.*User* mengisi data manual Delay:" "	Menetapkan pilihan kecepatan	Sistem men-*check* kecepatan pilihan *user*	Sesuai

4.2.3.4. Pengujian Mengatur Tempat Penyimpanan Gambar Sampel

Pengujian mengatur tempat penyimpanan gambar sampel diuji dengan pilihan *user* apakah alamat terpilih atau membatalkan pemilihan alamat. Pengaturan ini memilih *directory*, bukan memilih file. Pengujian mengatur tempat penyimpanan gambar sampel dapat dilihat pada Tabel 4.12.

Tabel 4.12 Pengujian Mengatur Tempat Penyimpanan Gambar Sampel

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* memilih ubahlokasi 2.*User* memilih lokasi 3.*User* menekan Ok	Menyimpan lokasi	Sistem menyimpan lokasi penyimpanan gambar	Sesuai
2	1.*User* memilih ubah lokasi 2.*User* menekan Cancel	Lokasi tidak ditentukan	Sistem membatalkan pemilihan lokasi	Sesuai

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
			penyimpanan gambar	

4.2.3.5. Pengujian Mengatur Kamera dan Port *Arduino*

Pengujian mengatur kamera dan port *Arduino* bertujuan untuk mengatur kamera dan port *Arduino* yang akan dipakai pada aplikasi. Pengaturan kamera dan port *Arduino* dipilih melalui menu *dropdown*, yang akan secara automatis akan mencari perangkat kamera dan *Arduino* yang tersedia ketika aplikasi pertama kali berjalan. Pemilihan kamera dan *port Arduino* adalah sebagain besar tanggung jawab dari *user*, karena sistem tidak dapat mengenali yang mana kamera dan *port Arduino* yang akan digunakan dalam alat bantu. Sistem hanya menampilkan pilihan dari beberapa kamera atau *port Arduino* saja. Pengujian memilih kamera dapat dilihat pada Tabel 4.13 dan pengujian memilih port *Arduino* dapat dilihat pada Tabel 4.14.

Tabel 4.13 Pengujian Memilih Kamera

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* memilih kamera (Asumsi kamera ada kamera tersambung)	Menyimpan Kamera dan menggunakan kamera	Sistem menyimpan dan menggunakan kamera	Sesuai
2	1.*User* memilih ubah kamera (Asumsi kamera tidak ada kamera tersambung)	Menampilkan pesan kesalahan	Sistem menampilkan pesan kesalahan tidak ada kamera yang terkoneksi	Sesuai

Tabel 4.14 Pengujian Memilih Port *Arduino*

No	Data Masukan	Hasil yang diharapkan	Hasil Pengamatan	Hasil Pengujian
1	1.*User* memilih *port Arduino* (Asumsi *port Arduino* ada *port Arduino* tersambung)	Menyimpan *Port Arduino* dan menggunakan *port Arduino*	Sistem menyimpan dan menggunakan *port Arduino*	Sesuai
2	1.*User* memilih ubah *port Arduino* (Asumsi *port Arduino* tidak ada *port Arduino* tersambung)	Menampilkan pesan kesalahan	Sistem menampilkan pesan kesalahan tidak ada *port Arduino* yang terkoneksi	Sesuai

4.2.4. Pengujian White Box

Pengujian *White Box* merupakan pengujian yang didasarkan pada pengecekan terhadap detail perancangan dengan menggunakan stuktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Tujuan dari pengujian *White Box* untuk mengetahui cara kerja suatu aplikasi atau perangkat lunak secara internal dan untuk menjamin operasi-operasi internal sesuai dengan spesifikasi yang telah ditetapkan dengan menggunakan struktur kontrol dari prosedur yang dirancang. Berdasarkan skenario pengujian, maka dapat dilakukan pengujian *White Box* pada aplikasi pengenalan objek sebagai berikut :

4.2.4.1. Pengujian Deteksi Petak

Pengujian deteksi petak ini dibagi menjadi beberapa tahap. Berikut ini adalah *pseudecode* tahap 1 pada Gambar 4.1 untuk mendeteksi petak papan permainan monopoli :

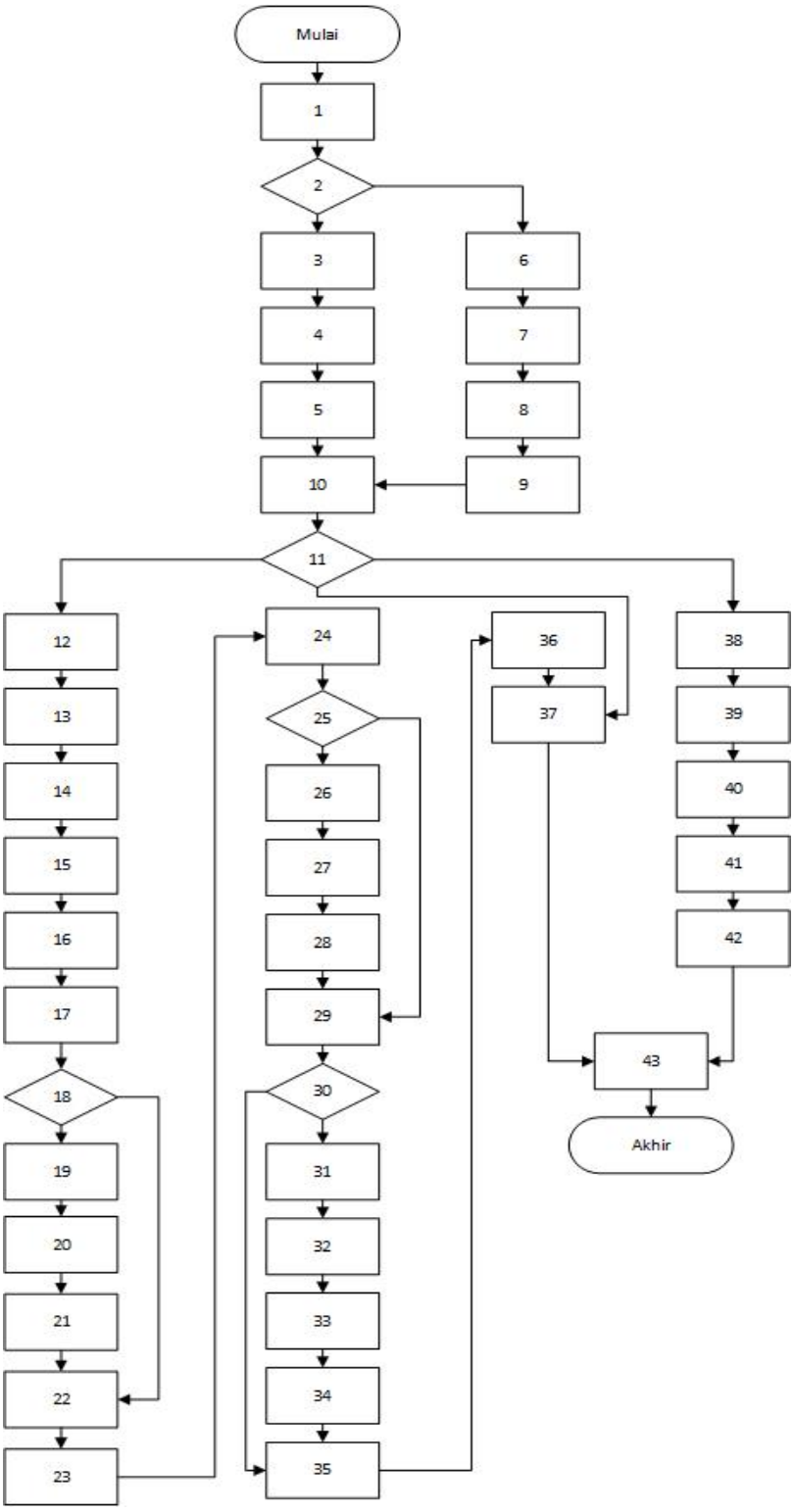
```
private void perform_SURF_detection_and_update_GUI(object sender, EventArgs e)
```

```
{
try
            {
                ImgSceneColor = CapWebcam.QueryFrame();
            }
            catch (Exception ex)
            {
                MessageBox.Show("" + ex.Message);
                return;
            }
            if (ImgSceneColor == null)
            {
                MessageBox.Show("Terdapat kesalahan dalam koneksi kamera.\n Mencoba mengkoneksi kamera", "Peringatan", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                Application.Idle -= new EventHandler(this.perform_SURF_detection_and_update_GUI);
                List<KeyValuePair<int, string>> ListCamerasData = new List<KeyValuePair<int, string>>();
                _SystemCamereas = DsDevice.GetDevicesOfCat(FilterCategory.VideoInputDevice);
                int _DeviceIndex = 0;
                foreach (DirectShowLib.DsDevice _Camera in _SystemCamereas)
                {
                    ListCamerasData.Add(new KeyValuePair<int, string>(_DeviceIndex, _Camera.Name));
                    _DeviceIndex++;
                }
                comboBox_kamera.DataSource = null;
                comboBox_kamera.Items.Clear();
                comboBox_kamera.DataSource = new BindingSource(ListCamerasData, null);
                comboBox_kamera.DisplayMember = "Value";
                comboBox_kamera.ValueMember = "Key";
                KeyValuePair<int, string> SelectedItem = (KeyValuePair<int, string>)comboBox_kamera.SelectedItem;
                _CameraIndex = SelectedItem.Key;
if (CapWebcam != null)
                {
                    Application.Idle -= new EventHandler(this.perform_SURF_detection_and_update_GUI);
                    CapWebcam.Dispose();
                    load_camera();
                }
                return;
```

```
37              }
38  else if (ImgToFindColor == null)
39              {
40                  imageBox_result.Image = ImgSceneColor;
41                  return;
42              }
43  SURFDetector surfdetector = new SURFDetector(500, false);
```

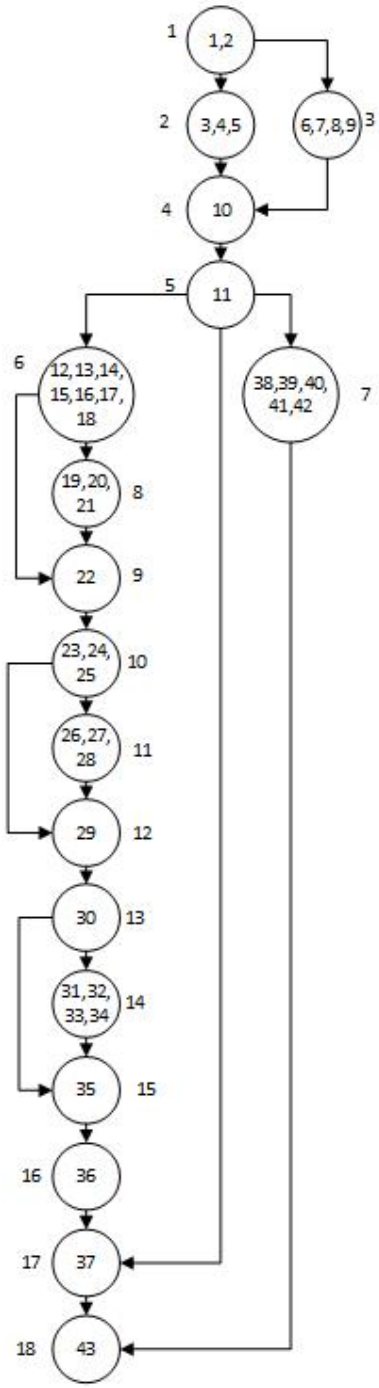
Gambar 4.1 *Pseudecode* Deteksi Petak Tahap 1

Berdasarkan *Pseudecode* deteksi petak tahap 1 pada Gambar 4.1, maka berikut adalah *flowchart* deteksi petak tahap 1 pada Gambar 4.2.



Gambar 4.2 *Flowchart* Deteksi Petak Tahap 1

Berdasarkan *flowchart* deteksi petak tahap 1 pada gambar Gambar 4.2, maka berikut ini adalah *flowgraph* deteksi petak tahap 1 pada Gambar 4.3.



Gambar 4.3 *Flowgraph* Deteksi Petak Tahap 1

Berdasarkan *flowgraph* deteksi petak tahap 1 pada Gambar 4.3, maka berikut ini adalah *cyclomatic complexity* deteksi petak tahap 1

E (jumlah busur) = 23

N (jumlah simpul) = 18

V(G) = E – N + 2

V(G) = 23 – 18 + 2

V(G) = 7

Setelah *cyclomatic complexity* diketahui nilainya yaitu 7, maka dapat dihitung *independent path* sebagai berikut :

Path 1 = 1 – 2 – 3 – 4 – 5 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 43

Path 2 = 1 – 2 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 43

Path 3 =1 – 2 – 3 – 4 – 5 – 10 – 11 – 37 – 43

Path 4 = 1 – 2 – 3 – 4 – 5 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 43

Path 5 = 1 – 2 – 3 – 4 – 5 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 43

Path 6 = 1 – 2 – 3 – 4 – 5 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 35 – 36 – 37 – 43

Path 7 = 1 – 2 – 3 – 4 – 5 – 10 – 11 – 38 – 39 – 40 – 41 – 42 – 43

Graph Matrixnya adalah sebagai Tabel 4.15 berikut:

Tabel 4.15 Graph Matrik Tahap 1

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1		1	1															
2				1														
3				1														
4					1													

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5						1	1										1	
6								1	1									
7																		1
8									1									
9										1								
10											1	1						
11												1						
12													1					
13														1	1			
14															1			
15																1		
16																	1	
17																		1
18																		

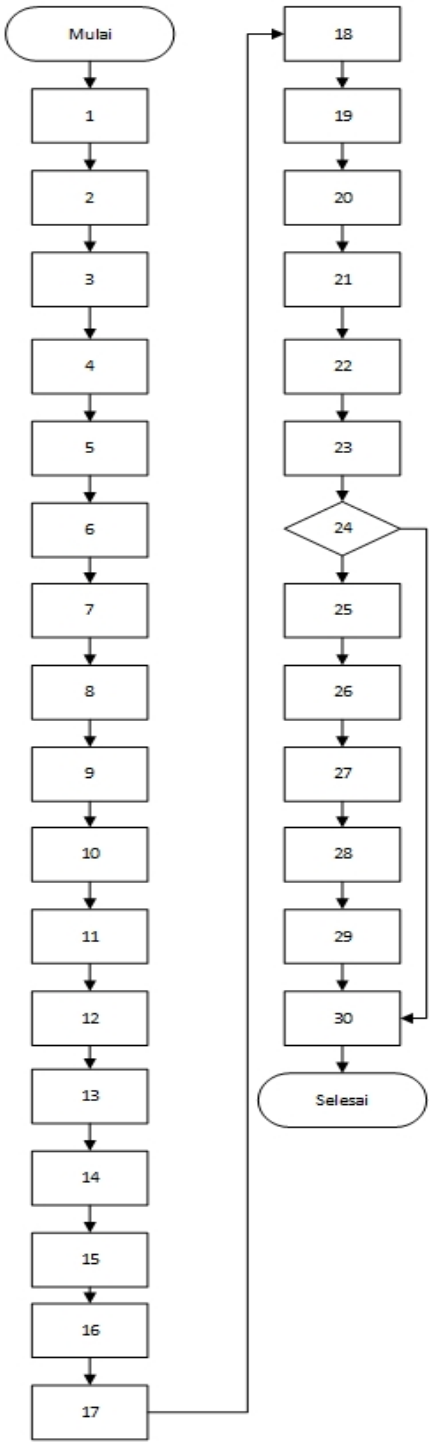
1. 1-1 =1
2. 1-1 =0
3. 1-1 =0
4. 1-1 =0
5. 3-1 =2
6. 2-1 =1
7. 1-1 =0
8. 1-1 =0
9. 1-1 =0
10. 2-1 =1
11. 1-1 =0
12. 1-1 =0
13. 2-1 =1
14. 1-1 =0
15. 1-1 =0
16. 1-1 =0

Cyclomatic Complexity dari *Graph Matrix* pada Tabel 4.15 adalah 6 + 1 = 7

```
1   Image<Gray, byte> imgSceneGray = null;
2   Image<Gray, byte> imgToFindGray = null;
3   VectorOfKeyPoint vkpSceneKeyPoints;
4   VectorOfKeyPoint vkpToFindKeyPoints;
5   Matrix<Single> mtxSceneDescriptors;
6   Matrix<Single> mtxToFindDescriptors;
7   Matrix<int> mtxMatchIndices;
8   Matrix<Single> mtxDistance;
9   Matrix<byte> mtxMask;
10  HomographyMatrix homographyMatrix = null;
11  int intKNumNearestNeighbors = 2;
12  int intNumNonZeroElements;
13  Rectangle rectImageToFind = new Rectangle(0, 0, 0, 0);
14  PointF[] ptfPointsf;
15  Point[] ptPoints;
16  imgSceneGray = ImgSceneColor.Convert<Gray, byte>();
17  imgToFindGray = ImgToFindColor.Convert<Gray, byte>();
18  vkpSceneKeyPoints = surfdetector.DetectKeyPointsRaw(imgSceneGray, null);
19  mtxSceneDescriptors = surfdetector.ComputeDescriptorsRaw(imgSceneGray, null, vkpSceneKeyPoints);
20  vkpToFindKeyPoints = surfdetector.DetectKeyPointsRaw(imgToFindGray, null);
21  mtxToFindDescriptors = surfdetector.ComputeDescriptorsRaw(imgToFindGray, null, vkpToFindKeyPoints);
22  Emgu.CV.FLANN.Index fln = new Index(mtxToFindDescriptors);
23  if (mtxSceneDescriptors != null)
24      {
25          mtxMatchIndices = new Matrix<int>(mtxSceneDescriptors.Rows, intKNumNearestNeighbors);
26          mtxDistance = new Matrix<Single>(mtxSceneDescriptors.Rows, intKNumNearestNeighbors);
27          fln.KnnSearch(mtxSceneDescriptors, mtxMatchIndices, mtxDistance, intKNumNearestNeighbors, 12);
28          mtxMask = new Matrix<byte>(mtxDistance.Rows, 1);
29          mtxMask.SetValue(255);
30      }
```

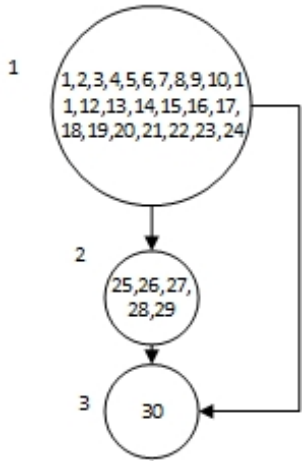
Gambar 4.4 *Pseudecode* Deteksi Petak Tahap 2

Berdasarkan *pseudecode* deteksi petak tahap 2 pada Gambar 4.4, maka berikut ini adalah *flowchart* deteksi petak tahap 2 pada Gambar 4.5.



Gambar 4.5 *FlowChart* Deteksi Petak Tahap 2

Berdasarkan *flowchart* deteksi petak tahap 2 pada Gambar 4.5, maka berikut ini adalah *flowgraph* deteksi petak tahap 2 pada Gambar 4.6.



Gambar 4.6 *FlowGraph* Deteksi Petak Tahap 2

Berdasarkan *flowgraph* deteksi petak pada Gambar 4.6, maka berikut ini adalah *cyclomatic complexity*

E (jumlah busur) = 3

N (jumlah simpul) = 3

V(G) = E – N + 2

V(G) = 3 – 3 + 2

V(G) = 2

Setelah *cyclomatic complexity* diketahui nilainya yaitu 2, maka dapat dihitung *independent path* sebagai berikut :

Path 1 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30

Path 2 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 30

Graph Matrixnya adalah sebagai Tabel 4.16 berikut:

Tabel 4.16 Graph Matrik Tahap 2

Node	1	2	3
1	1		1
2		1	
3			1

1. 2-1 = 1
2. 1-1=0
3. 1-1=0

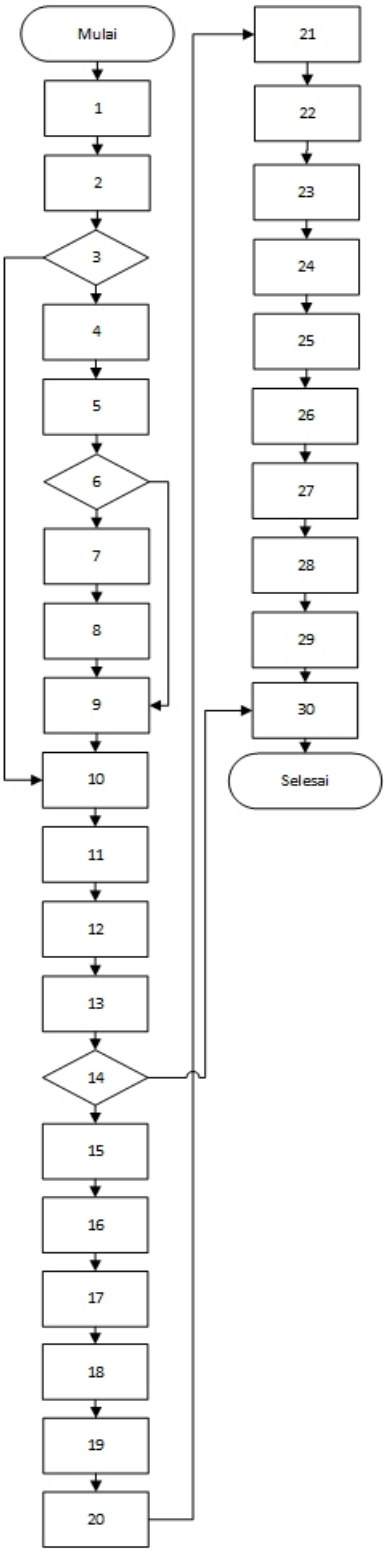
Cyclomatic Complexity adalah 1 + 1 = 2

```
1  Features2DToolbox.VoteForUniqueness(mtxDistance, 0.65, mtxMask);
2                  intNumNonZeroElements = CvInvoke.cvCountNonZero(mtxMask);
3                  if (intNumNonZeroElements >= 4)
4                  {
5                      intNumNonZeroElements = Features2DToolbox.VoteForSizeAndOrientation(vkpToFindKeyPoints, vkpSceneKeyPoints, mtxMatchIndices, mtxMask, 1.5, 20);
6                      if (intNumNonZeroElements >= 4)
7                      {
8                          homographyMatrix = Features2DToolbox.GetHomographyMatrixFromMatchedFeatures(vkpToFindKeyPoints, vkpSceneKeyPoints, mtxMatchIndices, mtxMask, 2);
9                      }
10                 }
11                 ImgCopyOfImageToFindWIthBorder = ImgToFindColor.Copy();
12                 ImgCopyOfImageToFindWIthBorder.Draw(new Rectangle(1, 1, ImgCopyOfImageToFindWIthBorder.Width - 3, ImgCopyOfImageToFindWIthBorder.Height - 3), BgrFoundImageColor, 2);
13                 ImgResult = ImgSceneColor;
14                 if (homographyMatrix != null)
15                 {
16                     rectImageToFind.X = 0;
17                     rectImageToFind.Y = 0;
18                     rectImageToFind.Width = imgToFindGray.Width;
19                     rectImageToFind.Height = imgToFindGray.Height;
20                     ptfPointsf = new PointF[] { new PointF(rectImageToFind.Left, rectImageToFind.Top), new PointF(rectImageToFind.Right, rectImageToFind.Top), new
```

```
    PointF(rectImageToFind.Right,    rectImageToFind.Bottom),    new
    PointF(rectImageToFind.Left, rectImageToFind.Bottom) };
21                  homographyMatrix.ProjectPoints(ptfPointsf);
22                  ptPoints      =      new      Point[]      {
    Point.Round(ptfPointsf[0]),          Point.Round(ptfPointsf[1]),
    Point.Round(ptfPointsf[2]), Point.Round(ptfPointsf[3]) };
23                  ImgResult.DrawPolyline(ptPoints,       true,
    BgrFoundImageColor, 10);
24                  MCvFont           f           =           new
    MCvFont(FONT.CV_FONT_HERSHEY_COMPLEX, 1.0, 1.0);
25                  ImgResult.Draw("X="                          +
    Point.Round(ptfPointsf[0]),          ref          f,          new
    Point(Point.Round(ptfPointsf[0]).X,
    Point.Round(ptfPointsf[0]).Y), new Bgr(17, 255, 57));
26                  ImgResult.Draw("Y="                          +
    Point.Round(ptfPointsf[1]),          ref          f,          new
    Point(Point.Round(ptfPointsf[1]).X,
    Point.Round(ptfPointsf[1]).Y), new Bgr(17, 255, 57));
27                  ImgResult.Draw("A="                          +
    Point.Round(ptfPointsf[2]),          ref          f,          new
    Point(Point.Round(ptfPointsf[2]).X,
    Point.Round(ptfPointsf[2]).Y), new Bgr(17, 255, 57));
28                  ImgResult.Draw("B="                          +
    Point.Round(ptfPointsf[3]),          ref          f,          new
    Point(Point.Round(ptfPointsf[3]).X,
    Point.Round(ptfPointsf[3]).Y), new Bgr(17, 255, 57));
29                  imageBox_result.Image = ImgResult;
30              }
```

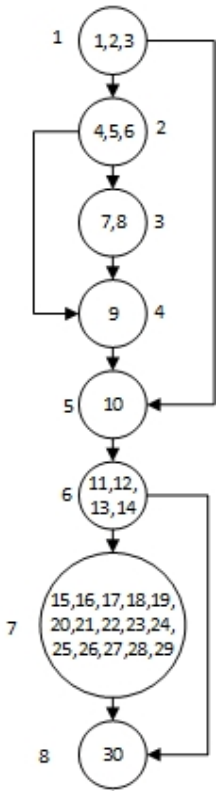
Gambar 4.7 *Pseudecode* Deteksi Petak Tahap 3

Berdasarkan *pseudecode* deteksi petak tahap 3 pada Gambar 4.7, maka berikut ini adalah *flowgraph* deteksi petak tahap 3 pada Gambar 4.8.



Gambar 4.8 *Flowchart* Deteksi Petak Tahap 3

Berdasarkan *flowchart* deteksi petak tahap 3 pada Gambar 4.8, maka berikut ini adalah *flowgraph* deteksi petak tahap 3 pada Gambar 4.9.



Gambar 4.9 *Flowgraph* Deteksi Petak Tahap 3

Berdasarkan *flowgraph* deteksi petak pada Gambar 4.9, maka berikut ini adalah *cyclomatic complexity*

E (jumlah busur) = 10
N (jumlah simpul) = 8
V(G) = E – N + 2
V(G) = 10 – 8 + 2
V(G) = 4

Setelah *cyclomatic complexity* diketahui nilainya yaitu 4, maka dapat dihitung *independent path* sebagai berikut :

Path 1 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30

Path 2 = 1 – 2 – 3 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30

Path 3 = 1 – 2 – 3 – 4 – 5 – 6 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30

Path 4 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 30

Graph Matrixnya adalah sebagai tabel Tabel 4.17 berikut:

Tabel 4.17 Graph Matrik Tahap 3

Node	1	2	3	4	5	6	7	8
1		1			1			
2			1	1				
3				1				
4					1			
5						1		
6							1	1
7								1
8								

1. 2-1 = 1
2. 2-1=1
3. 1-1=0
4. 1-1=0
5. 1-1=0
6. 2-1=1
7. 1-1=0

Cyclomatic Complexity adalah 3 + 1 = 4

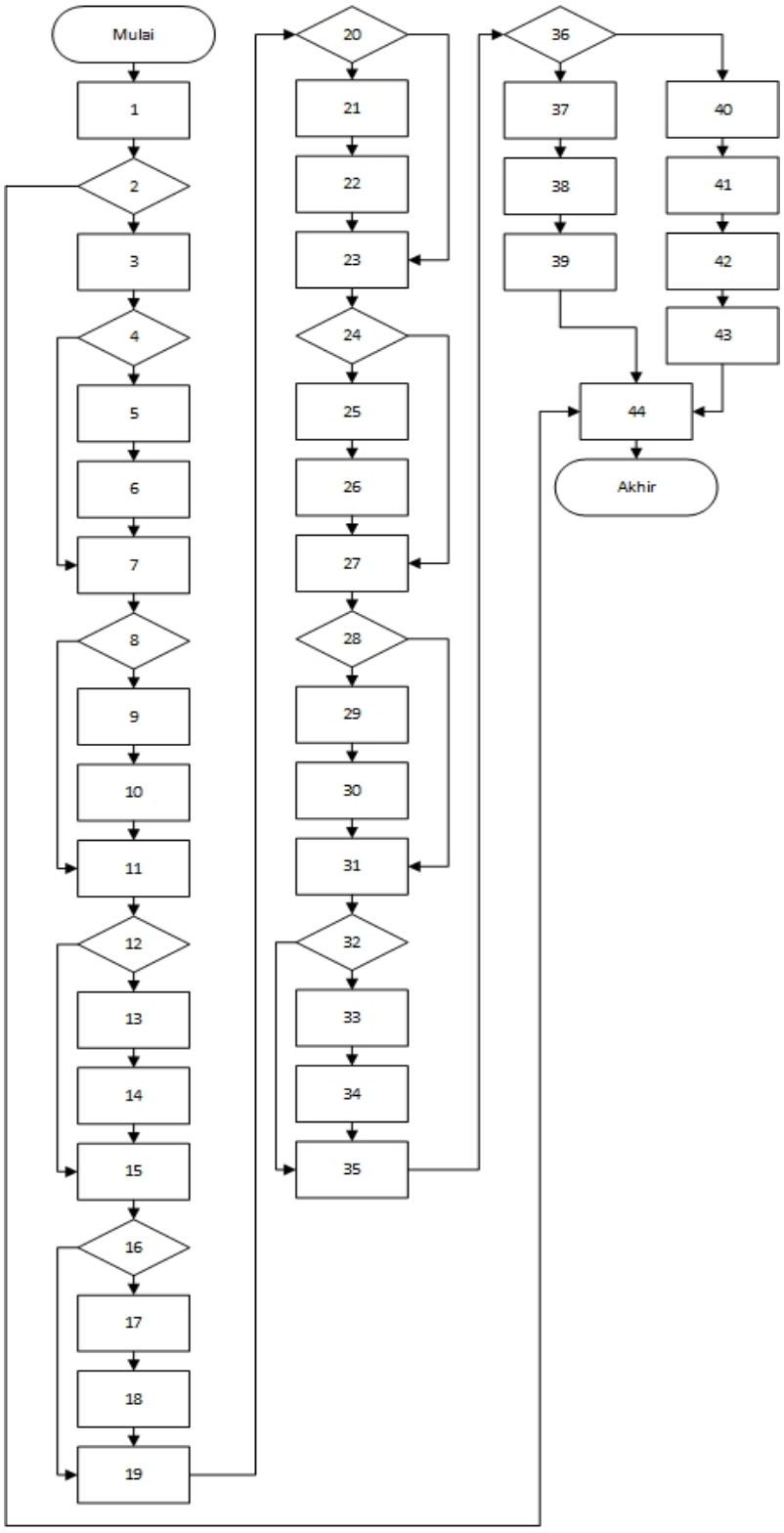
```
1  counter++;
2  if (counter >= 2)
3  {
4  if ((Convert.ToDouble(Point.Round(ptfPointsf[0]).X) >= temp[0] -
```

```
(temp[0] * 0.5)) &&
(Convert.ToDouble(Point.Round(ptfPointsf[0]).X) <= temp[0] +
(temp[0] * 0.5)) &&
Convert.ToDouble(Point.Round(ptfPointsf[0]).X) != 0)
{
    X1 = true;
}
if
((Convert.ToDouble(Point.Round(ptfPointsf[1]).X) >= temp[1] -
(temp[1] * 0.5)) &&
(Convert.ToDouble(Point.Round(ptfPointsf[1]).X) <= temp[1] +
(temp[1] * 0.5)) &&
Convert.ToDouble(Point.Round(ptfPointsf[1]).X) != 0)
{
    X2 = true;
}
if
((Convert.ToDouble(Point.Round(ptfPointsf[2]).X) >= temp[2] -
(temp[2] * 0.5)) &&
(Convert.ToDouble(Point.Round(ptfPointsf[2]).X) <= temp[2] +
(temp[2] * 0.5)) &&
Convert.ToDouble(Point.Round(ptfPointsf[2]).X) != 0)
{
    X3 = true;
}
if
((Convert.ToDouble(Point.Round(ptfPointsf[3]).X) >= temp[3] -
(temp[3] * 0.5)) &&
(Convert.ToDouble(Point.Round(ptfPointsf[3]).X) <= temp[3] +
(temp[3] * 0.5)) &&
Convert.ToDouble(Point.Round(ptfPointsf[3]).X) != 0)
{
    X4 = true;
}
if
((Convert.ToDouble(Point.Round(ptfPointsf[0]).Y) >= temp2[0] -
(temp2[0] * 0.5)) &&
(Convert.ToDouble(Point.Round(ptfPointsf[0]).Y) <= temp2[0] +
(temp2[0] * 0.5)) &&
Convert.ToDouble(Point.Round(ptfPointsf[0]).Y) != 0)
{
    Y1 = true;
}
if
((Convert.ToDouble(Point.Round(ptfPointsf[1]).Y) >= temp2[1] -
(temp2[1] * 0.5)) &&
(Convert.ToDouble(Point.Round(ptfPointsf[1]).Y) <= temp2[1] +
(temp2[1] * 0.5)) &&
Convert.ToDouble(Point.Round(ptfPointsf[1]).Y) != 0)
{
    Y2 = true;
```

```
27                          }
28                          if ((Convert.ToDouble(Point.Round(ptfPointsf[2]).Y) >= temp2[2] - (temp2[2] * 0.5)) && (Convert.ToDouble(Point.Round(ptfPointsf[2]).Y) <= temp2[2] + (temp2[2] * 0.5)) && Convert.ToDouble(Point.Round(ptfPointsf[2]).Y) != 0)
29                          {
30                              Y3 = true;
31                          }
32                          if ((Convert.ToDouble(Point.Round(ptfPointsf[3]).Y) >= temp2[3] - (temp2[3] * 0.5)) && (Convert.ToDouble(Point.Round(ptfPointsf[3]).Y) <= temp2[3] + (temp2[3] * 0.5)) && Convert.ToDouble(Point.Round(ptfPointsf[3]).Y) != 0)
33                          {
34                              Y4 = true;
35                          }
36                          if (X1 == true && X2 == true && X3 == true && X4 == true && Y1 == true && Y2 == true && Y3 == true && Y4 == true)
37                          {
38                              gambarstabil = true;
39                          }
40                          else
41                          {
42                              gambarstabil = false;
43                          }
44                      }
```

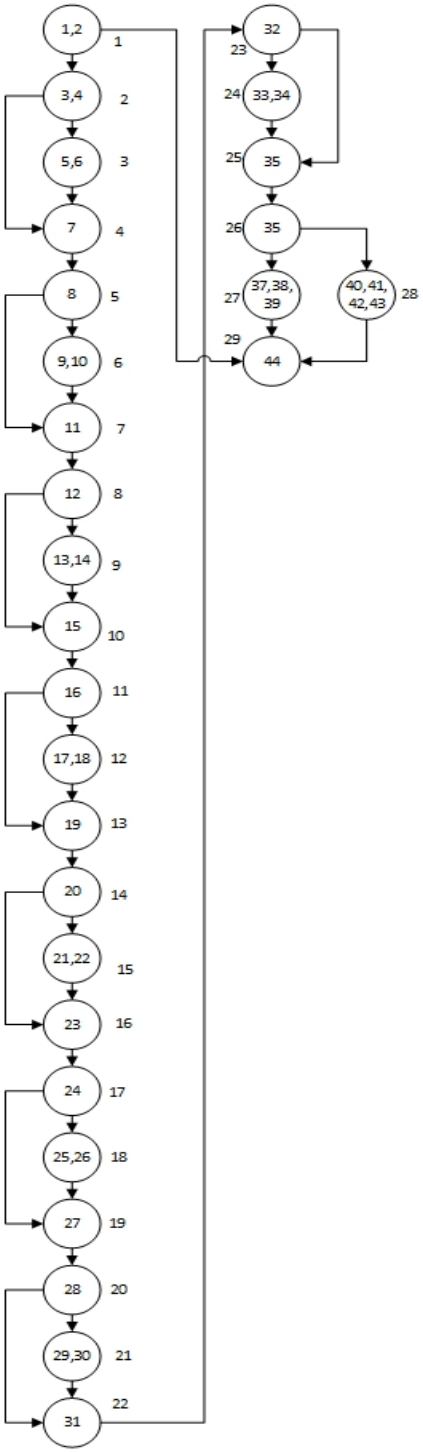
Gambar 4.10 *Pseudecode* Deteksi Petak Tahap 4

Berdasarkan *pseudecode* deteksi petak tahap 4 pada Gambar 4.10, maka berikut ini adalah *flowchart* deteksi petak tahap 4 pada Gambar 4.11.



Gambar 4.11 ***Flowchart*** **Deteksi Petak Tahap 4**

Berdasarkan *flowchart* deteksi petak tahap 4 pada Gambar 4.11, maka berikut ini adalah *flowgraph* deteksi petak tahap 4 pada Gambar 4.12.



Gambar 4.12 ***Flowgraph*** **Deteksi Petak Tahap 4**

Berdasarkan *flowgraph* deteksi petak tahap 4 pada Gambar 4.12, maka berikut ini adalah *cyclomatic complexity*

E (jumlah busur) = 38

N (jumlah simpul) = 29

V(G) = E – N + 2

V(G) = 38 – 29 + 2

V(G) = 11

Setelah *cyclomatic complexity* diketahui nilainya yaitu 11, maka dapat dihitung *independent path* sebagai berikut :

Path 1 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 44

Path 2 = 1 – 2 – 44

Path 3 = 1 – 2 – 3 – 4 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 44

Path 4 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 44

Path 5 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 44

Path 6 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 44

Path 7 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 44

Path 8 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 44

Path 9 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 31 – 32 – 33 – 34 – 35 – 36 – 37 – 38 – 39 – 44

Path 10 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 35 – 36 – 37 – 38 – 39 – 44

Path 11 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 29 – 30 – 31 – 32 – 33 – 34 – 35 – 36 – 40 – 41 – 42 – 43 – 44

Graph Matrixnya adalah sebagai Tabel 4.18 berikut:

Tabel 4.18 Graph Matrik Tahap 4

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1		1																											1
2			1	1																									
3				1																									
4					1																								
5						1	1																						
6							1																						
7								1																					

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
8									1	1																			
9										1																			
10											1																		
11												1	1																
12													1																
13														1															
14															1	1													
15																1													
16																	1												
17																		1	1										
18																			1										
19																				1									
20																					1	1							
21																						1							
22																							1						
23																								1	1				

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
24																									1				
25																										1			
26																											1	1	
27																													1
28																													1
29																													

1. 2-1 = 1
2. 2-1=1
3. 1-1=0
4. 1-1=0
5. 2-1 = 1
6. 1-1=0
7. 1-1=0
8. 2-1 = 1
9. 1-1=0
10. 1-1=0
11. 2-1 = 1

12. 1-1=0
13. 1-1=0
14. 2-1 = 1
15. 1-1=0
16. 1-1=0
17. 2-1 = 1
18. 1-1=0
19. 1-1=0
20. 2-1 = 1
21. 1-1=0
22. 1-1=0
23. 2-1 = 1
24. 1-1=0
25. 1-1=0
26. 2-1 = 1
27. 1-1=0
28. 1-1=0

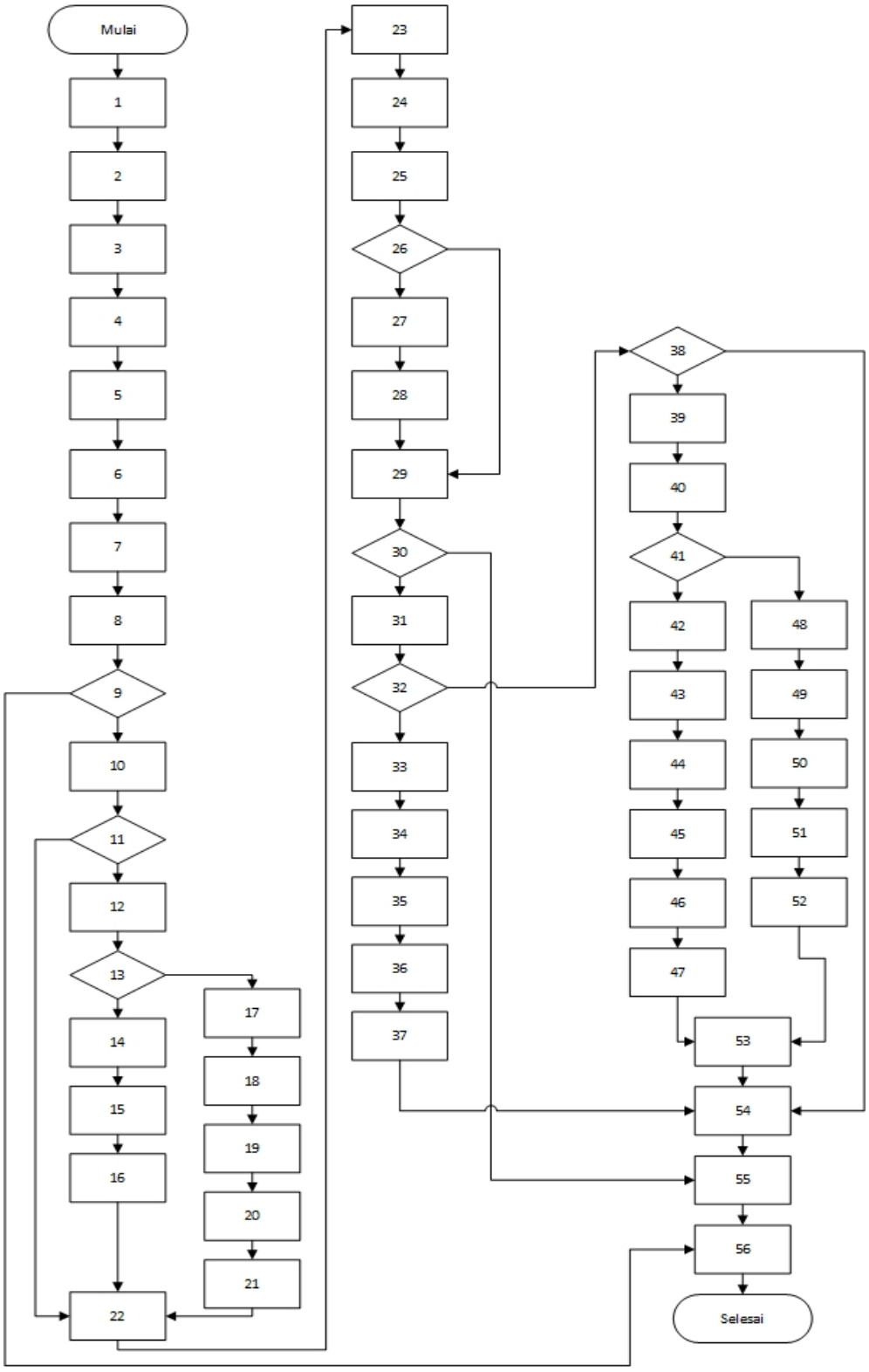
Cyclomatic Complexity adalah 10 + 1 = 11

```
1                           X1 = false;
2                           X2 = false;
3                           X3 = false;
4                           X4 = false;
5                           Y1 = false;
6                           Y2 = false;
7                           Y3 = false;
8                           Y4 = false;
9                           if (proses_pencocokan)
10                          {
11                              if (counter_hasil_pencocokan != 9)
12                              {
13                                  if (gambarstabil)
14                                  {
15  hasil_pencocokan[counter_hasil_pencocokan] = 1;
16                                  }
17                                  else
18                                  {
19  hasil_pencocokan[counter_hasil_pencocokan] = 0;
20                                  }
21                                  counter_hasil_pencocokan++;
22                              }
23                              else
24                              {
25                                  int hasil = 0;
26                                  for (int i = 0; i < 10; i++)
27                                  {
28                                      hasil     =     hasil     +
  hasil_pencocokan[i];
29                                  }
30                                  if (hasil <= 4)
31                                  {
32                                      if
  (proses_pencarian_petak_awal)
33                                      {
34  proses_akhir_kalibrasi(counter_kalibrasi_permainan + 1, false);
35  counter_kalibrasi_permainan = 0;
36  proses_pencarian_petak_awal = false;
37                                      }
38                                      else if (proses_kalibrasi)
```

```
39                                      {
40  counter_kalibrasi_permainan++;
41                                      if
    (counter_kalibrasi_permainan != 4)
42                                      {
43  kalibrasi_papan_proses();
44                                          counter = 0;
45  counter_hasil_pencocokan = 0;
46                                          return;
47                                      }
48                                      else
49                                      {
50  proses_akhir_kalibrasi(counter_kalibrasi_permainan + 1, false);
51  counter_kalibrasi_permainan = 0;
52                                      }
53                                  }
54                              }
55                          }
56                      }
```

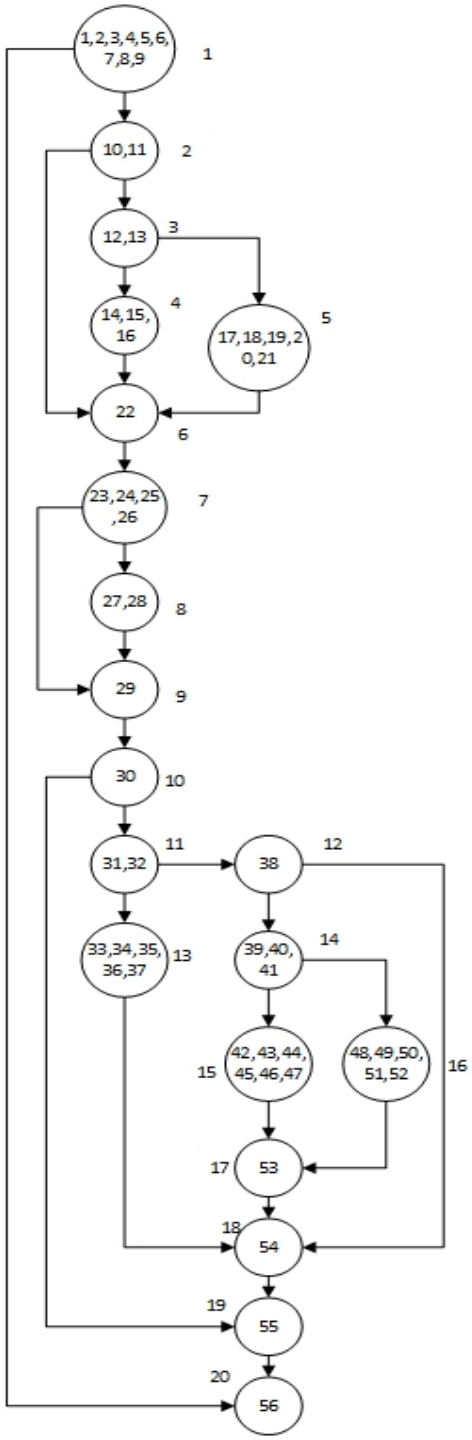
Gambar 4.13 *Pseudecode* Deteksi Petak Tahap 5

Berdasarkan *pseudecode* deteksi petak tahap 5 pada Gambar 4.13, maka berikut ini adalah *flowchart* deteksi petak tahap 5 pada Gambar 4.14.



Gambar 4.14 *Flowchart* Deteksi Petak Tahap 5

Berdasarkan *flowchart* deteksi petak tahap 5 pada Gambar 4.14, maka berikut ini adalah *flowgraph* deteksi petak tahap 5 pada Gambar 4.15.



Gambar 4.15 ***Flowgraph*** **Deteksi Petak Tahap 5**

Berdasarkan *flowgraph* deteksi petak tahap 5 pada Gambar 4.15, maka berikut ini adalah *cyclomatic complexity*

E (jumlah busur) = 27

N (jumlah simpul) = 20

V(G) = E – N + 2

V(G) = 27 – 20 + 2

V(G) = 9

Setelah *cyclomatic complexity* diketahui nilainya yaitu 9, maka dapat dihitung *independent path* sebagai berikut :

Path 1 = 1 – 2 – 3 – 4 – 6 – 7 – 8 – 9 – 10 – 11 – 13 – 18 – 19 – 20

Path 2 = 1 – 20

Path 3 = 1 – 2 – 3 – 4 – 6 – 7 – 8 – 9 – 10 – 11 – 13 – 18 – 19 – 20

Path 4 = 1 – 2 – 6 – 7 – 8 – 9 – 10 – 11 – 13 – 18 – 19 – 20

Path 5 = 1 – 2 – 3 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 13 – 18 – 19 – 20

Path 6 = 1 – 2 – 3 – 4 – 6 – 7 – 9 – 10 – 19 – 20

Path 7 = 1 – 2 – 3 – 4 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 14 – 15 – 17 – 18 – 19 – 20

Path 8 = 1 – 2 – 3 – 4 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 14 – 16 – 17 – 18 – 19 – 20

Path 9 = 1 – 2 – 3 – 4 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 18 – 19 – 20

Graph Matrixnya adalah sebagai Tabel 4.19 berikut:

Tabel 4.19 Graph Matrik Tahap 5

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

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

1. 2-1 = 1
2. 2-1=1
3. 2-1=1
4. 1-1=0
5. 1-1=0
6. 1-1=0
7. 2-1=1
8. 1-1=0
9. 1-1=0
10. 2-1=1
11. 2-1 = 1
12. 2-1=1
13. 1-1=0
14. 2-1 = 1
15. 1-1=0
16. 1-1=0
17. 1-1=0
18. 1-1=0
19. 1-1=0

Cyclomatic Complexity adalah 8 + 1 = 9

4.2.5. Pengujian Deteksi Petak pada Data Uji

Pengujian Deteksi petak pada data uji adalah pengujian aplikasi dalam mendeteksi petak menggunakan sekumpulan data uji, dengan cara melakukan perhitungan dalam pendeteksiannya. Untuk menghitung keakurasian dalam pengujian ini menggunakan rumus pada persamaan berikut. Diasumsikan pencahayaan menggunakan lampu belajar 11 watt. Pendeteksian petak dilakukan sebanyak 10 kali untuk memperkuat akurasi dari pendeteksian. Jika hasilnya kurang dari atau sama dengan 4 maka dinyatakan tidak terdeteksi, namun jika lebih atau sama dengan 5 maka dinyatakan terdeteksi. Pengujian deteksi petak dapat dilihat pada Tabel 4.20.

$$\text{Akurasi} = \frac{\text{Total Deteksi}}{\text{Jumlah percobaan} * 10} \text{x } 100\ \%$$

Tabel 4.20 Pengujian Deteksi Petak Menggunakan Data Uji

Percobaan	Data Uji Gambar			Kecepatan Deteksi Petak (s)										Petak Terdeteksi										Jumlah Deteksi	Hasil Deteksi
	Gambar	Gerakan Alat Bantu	Cahaya	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10		
1	Jernih	Tidak Goyang	Terang	1	1	1	1	1	1	1	1	1	1	Ya	Ya	Ya	Ya	Ya	Ya	Ya	Ya	Ya	Ya	10	Terdeteksi
2	Jernih	Goyang	Terang	1	1	1	1	1	1	1	1	1	1	Ya	Ya	Tidak	Ya	Ya	Ya	Ya	Ya	Ya	Ya	9	Terdeteksi
3	Noise	Tidak Goyang	Terang	1	1	1	1	1	1	1	1	1	1	Ya	Ya	Ya	Tidak	Ya	Tidak	Ya	Tidak	Ya	Ya	7	Terdeteksi
4	Noise	Goyang	Terang	1	1	1	1	1	1	1	1	1	1	Ya	Tidak	Tidak	Tidak	Ya	Ya	Tidak	Tidak	Ya	Ya	5	Terdeteksi

Akurasi = 31/(4*10) =31/40 =0.775 * 100% = 77.5%

Tingkat akurasi yang dihasilkan dalam berbagai macam kondisi menghasilkan akurasi sebesar 77.5%. Dari pengujian pada Tabel 4.20 diatas, dapat dibuktikan bahwa aplikasi dapat mendeteksi dengan baik apabila terdapat kondisi kondisi tertentu. Pada pengujian deteksi petak dengan kondisi cahaya yang gelap tidak menghasilkan deteksi petak yang baik sehingga hasil dari pendeteksian selalu tidak terdeteksi.

4.2.6. Pengujian Multiplayer

Pengujian *Multiplayer* adalah pengujian aplikasi untuk menjalankan deteksi kepemilikan petak terhadap beberapa pemain, dengan memanfaatkan jenis kartu yang dimiliki oleh pemain, serta jumlah pemain. Diamsusikan kartu yang dimiliki oleh pemain terbagi menjadi dua buah kartu, yaitu Kartu Biasa(KT) sebagai kartu yang tidak dicari oleh aplikasi dan Kartu Tujuan (KT) sebagai kartu yang dicari oleh aplikasi. Pengujian *Multiplayer* diuji dengan pemain lebih dari satu, kali ini pengujian akan dilakukan dengan 2 pemain. Tiap pemain memiliki 5 tempat kartu pemain. Gambar kartu dan petak permainan jernih atau bersih dari noise atau bercak pada gambar, pencahayaan menggunakan lampu belajar 11 Watt. Pengujiannya adalah sebagai Tabel 4.21 berikut.

$$\text{Akurasi} = \frac{\text{Jumlah Hasil Pengujian yang sesuai}}{\text{Jumlah pengujian}} \text{x } 100\ \%$$

Tabel 4.21 Pengujian *Multiplayer*

No	Jumlah Pemain	Kartu pada Kartu Properti Pemain										Kecepatan Deteksi (s)	Ditemukan di (Pemain, Kartu pemain)	Hasil Deteksi	Hasil Deteksi yang diharapkan	Hasil Pengujian
		1					2									
		1	2	3	4	5	1	2	3	4	5					
1	1	-	-	-	-	-	-	-	-	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
2	1	KT	-	-	-	-	-	-	-	-	-	5	1,1	Terdeteksi	Terdeteksi	Sesuai
3	1	-	KT	-	-	-	-	-	-	-	-	8	1,2	Terdeteksi	Terdeteksi	Sesuai
4	1	-	-	KT	-	-	-	-	-	-	-	10	1,3	Terdeteksi	Terdeteksi	Sesuai
5	1	-	-	-	KT	-	-	-	-	-	-	14	1,4	Terdeteksi	Terdeteksi	Sesuai
6	1	-	-	-	-	KT	-	-	-	-	-	18	1,5	Terdeteksi	Terdeteksi	Sesuai
7	1	KB	-	-	-	-	-	-	-	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
8	1	-	KB	-	-	-	-	-	-	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
9	1	-	-	KB	-	-	-	-	-	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
10	1	-	-	-	KB	-	-	-	-	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
11	1	-	-	-	-	KB	-	-	-	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
12	1	KB	KT	-	-	-	-	-	-	-	-	8	1,2	Terdeteksi	Terdeteksi	Sesuai

No	Jumlah Pemain	Kartu pada Kartu Properti Pemain										Kecepatan Deteksi (s)	Ditemukan di (Pemain, Kartu pemain)	Hasil Deteksi	Hasil Deteksi yang diharapkan	Hasil Pengujian
		1					2									
		1	2	3	4	5	1	2	3	4	5					
13	1	KB	KB	KT	-	-	-	-	-	-	-	10	1,3	Terdeteksi	Terdeteksi	Sesuai
14	1	KB	KB	KB	KT	-	-	-	-	-	-	14	1,4	Terdeteksi	Terdeteksi	Sesuai
15	1	KB	KB	KB	KB	KT	-	-	-	-	-	18	1,5	Terdeteksi	Terdeteksi	Sesuai
16	2	-	-	-	-	-	-	-	-	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
17	2	-	-	-	-	-	KT	-	-	-	-	5	2,1	Terdeteksi	Terdeteksi	Sesuai
18	2	-	-	-	-	-	-	KT	-	-	-	8	2,2	Terdeteksi	Terdeteksi	Sesuai
19	2	-	-	-	-	-	-	-	KT	-	-	10	2,3	Terdeteksi	Terdeteksi	Sesuai
20	2	-	-	-	-	-	-	-	-	KT	-	14	2,4	Terdeteksi	Terdeteksi	Sesuai
21	2	-	-	-	-	-	-	-	-	-	KT	18	2,5	Terdeteksi	Terdeteksi	Sesuai
22	2	-	-	-	-	-	KB	-	-	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
23	2	-	-	-	-	-	-	KB	-	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
24	2	-	-	-	-	-	-	-	KB	-	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
25	2	-	-	-	-	-	-	-	-	KB	-	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
26	2	-	-	-	-	-	-	-	-	-	KB	18	-	Tidak Terdeteksi	Tidak Terdeteksi	Sesuai
27	2	-	-	-	-	-	KB	KT	-	-	-	8	2,2	Terdeteksi	Terdeteksi	Sesuai
28	2	-	-	-	-	-	KB	KB	KT	-	-	10	2,3	Terdeteksi	Terdeteksi	Sesuai
29	2	-	-	-	-	-	KB	KB	KB	KT	-	14	2,4	Terdeteksi	Terdeteksi	Sesuai
30	2	-	-	-	-	-	KB	KB	KB	KB	KT	18	2,5	Terdeteksi	Terdeteksi	Sesuai

Akurasi = 30/30 *100% = 100%

Pengujian *Multiplayer* pada tabel diatas menghasilkan nilai akurasi 100%, pengujian ini juga sesuai dengan hasil yang diharapkan.

4.2.7. Kesimpulan Pengujian Alpha

Berdasarkan hasil pengujian sistem yang telah dilakukan secara keseluruhan, dapat ditarik kesimpulan bahwa secara fungsional aplikasi sudah dapat digunakan sesuai dengan harapan. Pengujian deteksi petak pada beberapa data uji menghasilkan nilai 77.5%, yang menyatakan bahwa aplikasi dapat berjalan dengan baik dengan kondisi tertentu. Serta pengujian *Multiplayer* yang menghasilkan nilai akurasi 100%, yang menyatakan bahwa aplikasi dapat berjalan sesuai dengan harapan yang diinginkan.