BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Setelah sistem dianalisis dan didesain secara rinci, maka selanjutnya akan menuju tahap implementasi. Implementasi sistem bertujuan untuk menjelaskan tentang manual modul kepada *user* yang akan menggunakan sistem, sehingga dapat memberikan masukan kepada pembangun sistem. Tahapan - tahapan implementasi tersebut berupa implementasi perangkat lunak, implementasi perangkat keras, implementasi antarmuka, implementasi program dan pengujian program.

4.1.1 Implementasi Perangkat Lunak

Implementasi Perangkat Lunak merupakan proses instalasi perangkat lunak, sehingga dapat beroperasi dengan benar. Proses implementasi perangkat lunak meliputi :

Tabel 4-1 Implementasi Perangkat Lunak

Perangkat Lunak	Keterangan
Microsoft Windows XP Professional	Sistem operasi
Microsoft Visual Studio 2010	Aplikasi pembangun sistem, pemodelan sistem
OpenCV 2.2	Library Tambahan

4.1.2 Implementasi Perangkat Keras

Implemantasi perangkat keras merupakan realisasi dari analisis dan perancangan kebutuhan perangkat keras. Implementasi perangkat keras yang dilakukan meliputi perangkat keras yang digunakan dalam pembangunan sistem dan perangkat keras yang diperuntukan bagi komputer yang akan menggunakan aplikasi ini.

Berikut ini merupakan spesifikasi perangkat keras untuk komputer dan juga *webcam* yang digunakan dalam pembangunan sistem :

Tabel 4-2 Implementasi Perangkat Keras Pembangun Sistem

Perangkat Keras	Spesifikasi Perangkat Keras
Komputer	- Sistem operasi Windows XP Profesisonal SP 3 - Prosessor AMD Athlon 3200+ 2.01 GHz - RAM 1 GB - VGA ATI Radeon X550 256 MB
WebCam	-Logitech C120 1,3 MP 640 x 480

Berikut ini merupakan spesifikasi perangkat keras untuk komputer dan juga *webcam* yang akan digunakan untuk menjalankan aplikasi :

Tabel 4-3 Implementasi perangkat keras yang akan digunakan

Perangkat Keras	Spesifikasi Perangkat Keras
Komputer	- Sistem operasi : Windows XP - Prosessor Intel Pentium 3 or higher - RAM 256 MB or higher
WebCam	- Video capture minimal 320 x 240 - Resolusi minimal 1 MP

4.1.3 Implementasi Antarmuka Sistem

Dari perancangan antarmuka yang telah dibuat pada bab sebelumnya, maka tahap selanjutnya yaitu mengimplementasikannya menjadi sebuah tampilan. Implementasi antarmuka sistem meliputi :

Tabel 4-4 Implementasi antarmuka sistem

No	Nama Menu	Nama File	Keterangan
1.	Form Menu Utama	MenuUtama.cs	Tampilan utama aplikasi
2.	Form Pengaturan	Pengaturan.cs	Tampilan pengaturan kamera
3.	Petunjuk Penggunaan	Petunjuk.cs	Petunjuk penggunaan aplikasi
4.	Form Tentang	Tentang.cs	Tampilan profil pembuat aplikasi
5.	Pesan Keluar Aplikasi	-	Pesan validasi untuk keluar aplikasi

4.2 Pengujian Sistem

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, perancangan dan pengkodean.

4.2.1 Rencana Pengujian

Pengujian yang dilakukan ada dua tahapan yaitu pengujian *alpha* dan pengujian *beta*. Pengujian *alpha* digunakan untuk menguji sistem, metode yang digunakan pada pengujian *alpha* adalah metode *white-box*. Pengujian *white-box* digunakan untuk meyakinkan semua perintah dan kondisi dieksekusi secara minimal. Pengujian *beta* digunakan untuk memeriksa tingkat keakuratan sistem, pada pengujian *beta* akan dilakukan uji performansi sistem.

4.2.1.1 Pengujian *Alpha*

Pengujian *alpha* digunakan untuk menguji aplikasi dengan berbagai kondisi, metode yang digunakan pada pengujian *alpha* adalah metode *white-box*. Pengujian *white-box* digunakan untuk meyakinkan semua perintah dan kondisi pada perangkat lunak pengendali pointer agar dieksekusi secara minimal. Pengujian *white-box* yang pada perangkat lunak pengendali pointer menggunakan dua *tools* yaitu *Flow Graph* yang digunakan untuk menggambarkan alur dari algoritma dan *Graph Matriks* yang digunakan untuk menggenerasi *flow graph*. Adapun Pengujian *white-box* pada perangkat lunak pengendali pointer adalah sebagai berikut :

1. Pengujian Pembuatan Haar Feature

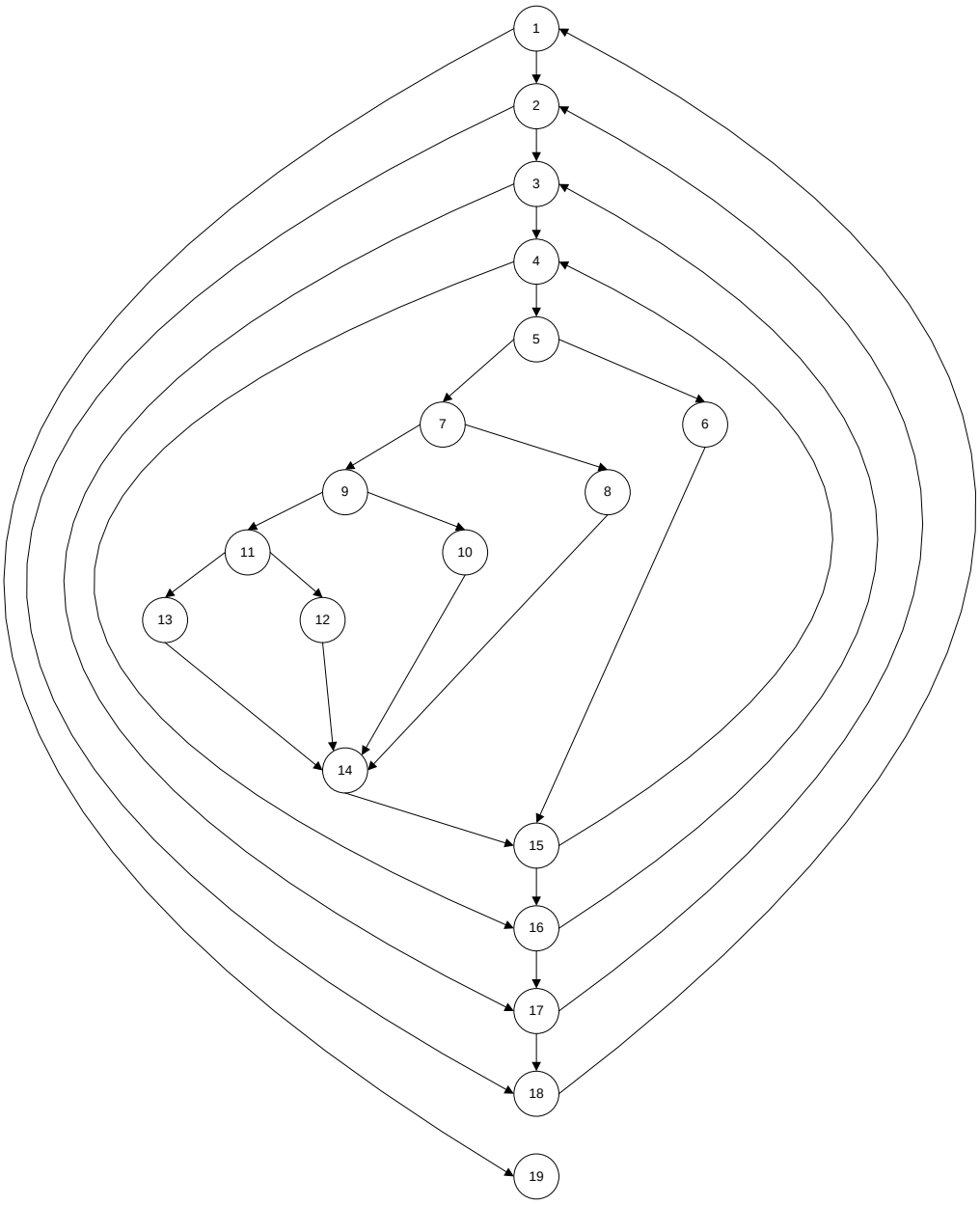
Berikut ini adalah algoritma pembuatan haar feature yang akan diuji :

```
1.    for( int x = 0; x < winSize.width; x++ )
      {
2.     for( int y = 0; y < winSize.height; y++ )
       {
3.      for( int dx = 1; dx <= winSize.width; dx++ )
        {
4.       for( int dy = 1; dy <= winSize.height; dy++ )
         {
5.        if ( (x+dx*2 <= winSize.width) && (y+dy <=
      winSize.height) )
           {
6.          features.push_back( Feature( offset, false,
      x, y, dx*2, dy, -1,x+dx, y, dx  , dy, +2 ) );
           }
7.         else if ( (x+dx*2 <= winSize.height) && (y+dy <=
      winSize.width) )
          {

8.          features.push_back( Feature( offset, false,
       y,x, dy, dx*2, -1,y, x+dx, dy, dx,   +2 ) );
            }
9.          else if ( (x+dx*3 <= winSize.width) && (y+dy <=
      winSize.height) )
            {
10.          features.push_back( Feature( offset, false,
      x,y, dx*3, dy, -1,x+dx, y, dx  , dy, +3 ) );
      }
11.         else if ( (x+dx*3 <= winSize.height) && (y+dy <=
      winSize.width) )
      {
12.         features.push_back( Feature( offset, false,
       y, x, dy, dx*3, -1,y, x+dx, dy, dx,   +3 ) );
      }
13.         else
      {
       features.push_back( Feature( offset, false,
       x,    y,    dx*2, dy*2, -1, x,    y,    dx,   dy,   +2,
       x+dx, y+dy, dx,   dy,   +2 ) );
14.       }
15.      }
16.    }
17.   }
18. }
      numFeatures = (int)features.size();
19. }
```

a. *Flow Graph* Pembuatan Haar Feature

Berikut ini adalah *flow graph* pembuatan haar feature :



Gambar 4.1 *Flow Graph* Pembuatan Haar feature

Keterangan :

◯ = Menggambarkan Kondisi

→ = Menggambarkan Aksi

b. *Graph Matriks* Pembuatan Haar Feature

Berikut ini adalah *graph matriks* dari *flow graph* pembuatan haar feature :

Tabel 4-5 *Graph Matriks* Pembuatan Haar Feature

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

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

2. Pengujian *Cascade Classifier*

Berikut ini adalah algoritma cascade classifier yang akan diuji :

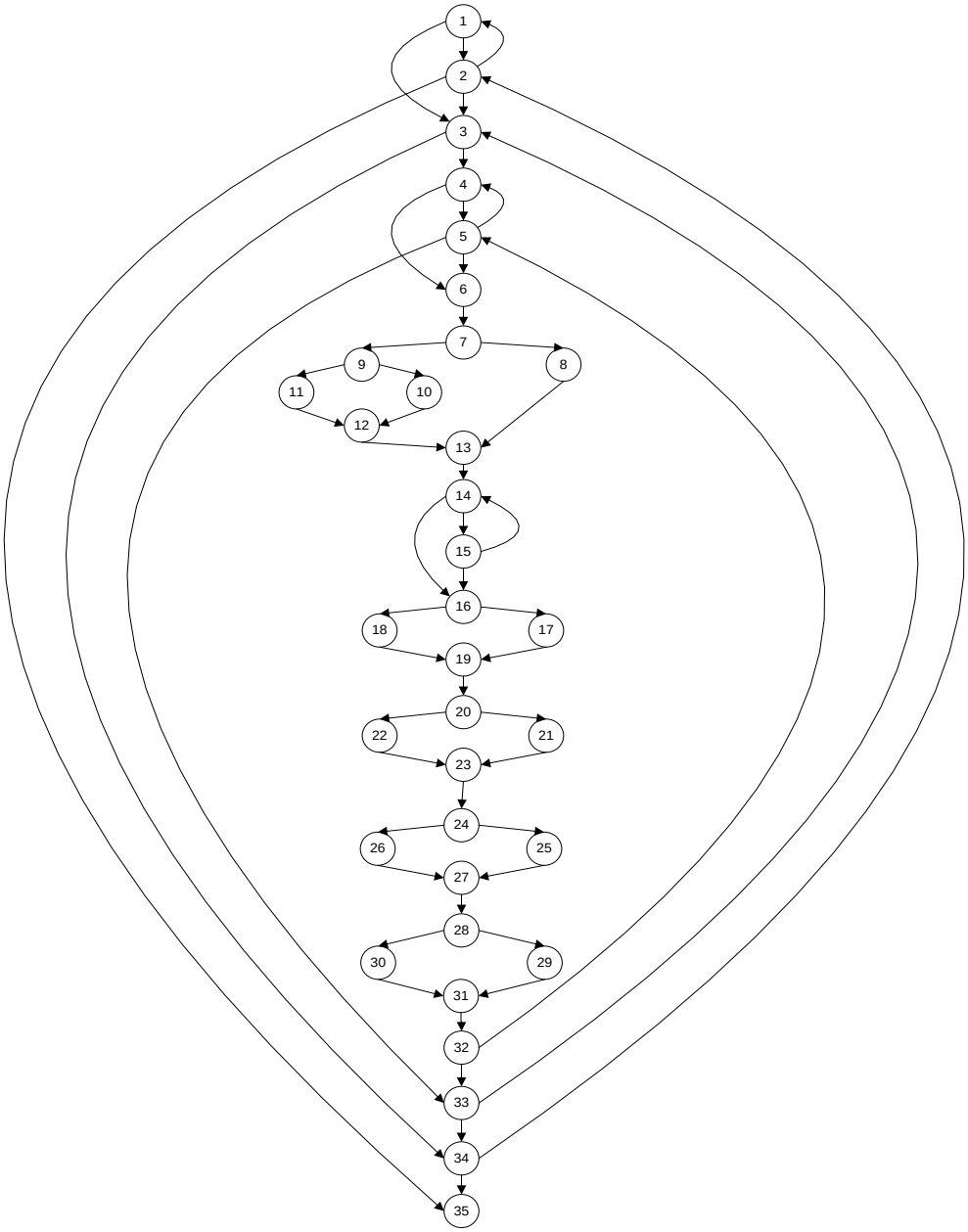
```
1.    for( i = 0; i < _cascade->count; i++ )
      {
2.     int j, k, l;
3.      for( j = 0; j < cascade->stage_classifier[i].count; j++ )
        {
4.       for( l = 0; l < cascade->
      stage_classifier[i].classifier[j].count;  l++ )
      {
5.     CvHaarFeature* feature = &_cascade->
       stage_classifier[i].classifier[j].haar_feature[l];
       CvHidHaarFeature* hidfeature = &cascade->
       stage_classifier[i].classifier[j].node[l].feature;
       CvRect r[3];
       int base_w = -1, base_h = -1;
       int new_base_w = 0, new_base_h = 0;
       int kx, ky;
       int flagx = 0, flagy = 0;
       int x0 = 0, y0 = 0;
       int nr;
6.     for( k = 0; k < CV_HAAR_FEATURE_MAX; k++ )
      {
7.        if( !hidfeature->rect[k].p0 )
8.        break;
      r[k] = feature->rect[k].r;
      base_w = (int)CV_IMIN( (unsigned)base_w,
      (unsigned)(r[k].width-1) );
      base_w = (int)CV_IMIN( (unsigned)base_w, (unsigned)(r[k].x -
      r[0].x-1) );
      base_h = (int)CV_IMIN( (unsigned)base_h,
      (unsigned)(r[k].height-1) );
      base_h = (int)CV_IMIN( (unsigned)base_h, (unsigned)(r[k].y -
      r[0].y-1) );
         }
      nr = k;
      base_w += 1;
      base_h += 1;
      kx = r[0].width / base_w;
      ky = r[0].height / base_h;

9.       if( kx <= 0 )
          {
10.        flagx = 1;
           new_base_w = cvRound( r[0].width * scale ) / kx;
           x0 = cvRound( r[0].x * scale );
          }
11.       else if( ky <= 0 )
         {
12.       flagy = 1;
          new_base_h = cvRound( r[0].height * scale ) / ky;
          y0 = cvRound( r[0].y * scale );
13.       }
```

```
14.     for( k = 0; k < nr; k++ )
       {
15.      CvRect tr;
       double correction_ratio;
16.      if( flagx )
       {
17.      tr.x = (r[k].x - r[0].x) * new_base_w / base_w + x0;
         tr.width = r[k].width * new_base_w / base_w;
       }
18.      else
       {
        tr.x = cvRound( r[k].x * scale );
        tr.width = cvRound( r[k].width * scale );
19.    }
20.     if( flagy )
        {
21.      tr.y = (r[k].y - r[0].y) * new_base_h / base_h + y0;
         tr.height = r[k].height * new_base_h / base_h;
        }
22.     else
       {
        tr.y = cvRound( r[k].y * scale );
        tr.height = cvRound( r[k].height * scale );
23.    }
24.     if( !feature->tilted )
       {
25.     hidfeature->rect[k].p0 = sum_elem_ptr(*sum, tr.y, tr.x);
        hidfeature->rect[k].p1 = sum_elem_ptr(*sum, tr.y, tr.x +
        tr.width);
        hidfeature->rect[k].p2 = sum_elem_ptr(*sum, tr.y +
        tr.height, tr.x);
      }
26.     else
        {
         hidfeature->rect[k].p2 = sum_elem_ptr(*tilted, tr.y +
         tr.width, tr.x + tr.width);
         hidfeature->rect[k].p0 = sum_elem_ptr(*tilted, tr.y,
         tr.x);
         hidfeature->rect[k].p1 = sum_elem_ptr(*tilted, tr.y +
         tr.height, tr.x - tr.height);
27.     }
        hidfeature->rect[k].weight = (float)(feature->
        rect[k].weight *   correction_ratio);
28.     if( k == 0 )
29.     area0 = tr.width * tr.height;
30.     else
        sum0 += hidfeature->rect[k].weight * tr.width * tr.height;
31.       }
        hidfeature->rect[0].weight = (float)(-sum0/area0);
32.      }
33.     }
34.    }
35.  }
```

a. *Flow Graph Cascade Classifier*

Berikut ini adalah *flow graph cascade classifier* :



Gambar 4.2 *Flow Graph Cascade Classifier*

Keterangan :

◯ = Menggambarkan Kondisi

→ = Menggambarkan Aksi

b. *Graph Matriks Cascade Classifier*

Berikut ini adalah *graph matriks* dari *flow graph cascade classifier :*

Tabel 4-6 *Graph Matriks Class classifier*

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

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

3. Pengujian *Integral Image*

Berikut ini adalah algoritma integral image yang akan diuji :

```
1.      if (image.cols < origWinSize.width || image.rows <
        origWinSize.height)
2.      return false;

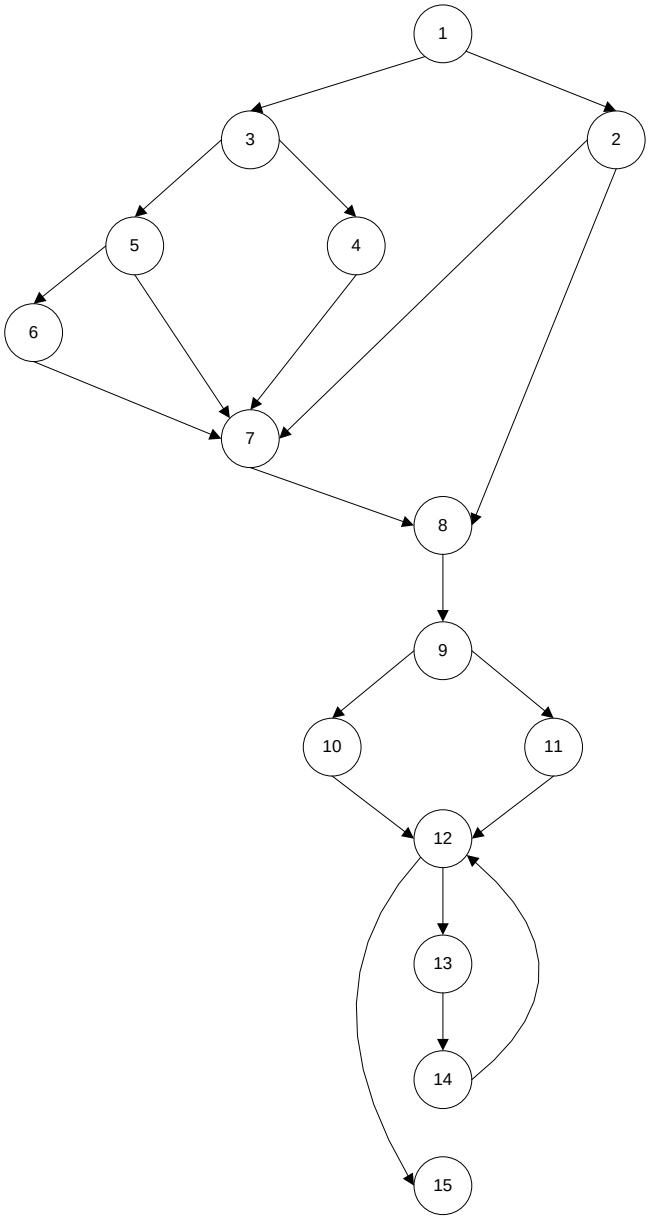
3.      else if( sum0.rows < rn || sum0.cols < cn )
4.      {
         sum0.create(rn, cn, CV_32S);
         sqsum0.create(rn, cn, CV_64F);
5.       if (hasTiltedFeatures)
6.       tilted0.create( rn, cn, CV_32S);
7.      }
8.      sum = Mat(rn, cn, CV_32S, sum0.data);
        sqsum = Mat(rn, cn, CV_32S, sqsum0.data);

9.        if( hasTiltedFeatures )
10.       {
           tilted = Mat(rn, cn, CV_32S, tilted0.data);
           integral(image, sum, sqsum, tilted);
          }
11.       else
           integral(image, sum, sqsum);
           const int* sdata = (const int*)sum.data;
           const double* sqdata = (const double*)sqsum.data;
           size_t sumStep = sum.step/sizeof(sdata[0]);
           size_t sqsumStep = sqsum.step/sizeof(sqdata[0]);
           CV_SUM_PTRS( p[0], p[1], p[2], p[3], sdata, normrect,
           sumStep );
           _SUM_PTRS( pq[0], pq[1], pq[2], pq[3], sqdata,
           normrect, sqsumStep );
           size_t fi, nfeatures = features->size();

12.     for( fi = 0; fi < nfeatures; fi++ )
13.     featuresPtr[fi].updatePtrs( !featuresPtr[fi].tilted ? sum :
        tilted );
14.     return true;
15.     }
```

a. *Flow Graph Integral Image*

Berikut ini adalah *flow graph Integral Image*:



Gambar 4.3 *Flow Graph Integral Image*

Keterangan :

◯ = Menggambarkan Kondisi

→ = Menggambarkan Aksi

b. *Graph Matriks Integral Image*

Berikut ini adalah *graph matriks* dari *flow graph integral image :*

Tabel 4-7 *Graph Matriks Integral Image*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		1	1												
2							1	1							
3				1	1										
4							1								
5						1	1								
6							1								
7								1							
8									1						
9										1	1				
10												1			
11												1			
12													1		1
13														1	
14												1			
15															

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

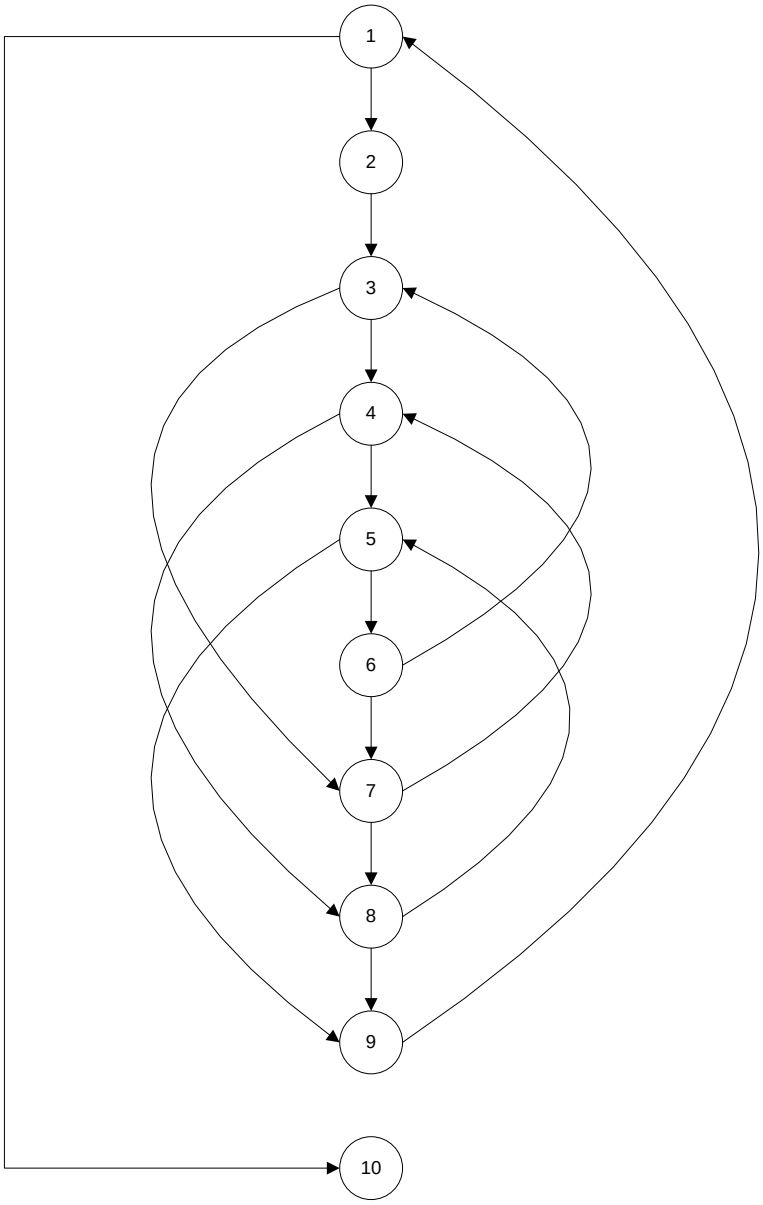
4. Pengujian *Feature Matching*

Berikut ini adalah algoritma feature matching yang akan diuji :

```
1.    while (true)
      {
2.    IplImage img = cap.QueryFrame();
      CvSeq<CvAvgComp> lefteye = Cv.HaarDetectObjects(img,
      cascadelefteye, Cv.CreateMemStorage(), ScaleFactor,
      MinNeighbors, HaarDetectionType.DoCannyPruning, MinSize);
      CvSeq<CvAvgComp> righteye = Cv.HaarDetectObjects(img,
      cascaderighteye, Cv.CreateMemStorage(), ScaleFactor,
      MinNeighbors, HaarDetectionType.DoCannyPruning, MinSize);
      CvSeq<CvAvgComp> faces = Cv.HaarDetectObjects(img, cascade,
      Cv.CreateMemStorage(), ScaleFactor, MinNeighbors,
      HaarDetectionType.DoCannyPruning, MinSize);
3.    foreach (CvAvgComp face in faces.AsParallel())
4.    foreach (CvAvgComp eye1 in lefteye.AsParallel())
5.    foreach (CvAvgComp eye2 in righteye.AsParallel())
      {
6.     img.Rectangle(face.Rect, CvColor.Red, 2,
       LineType.AntiAlias);
7.     img.Rectangle(eye1.Rect, CvColor.Blue, 2,
       LineType.AntiAlias);
8.     img.Rectangle(eye2.Rect, CvColor.Blue, 2,
       LineType.AntiAlias);
9.     }
10.   }
```

a. *Flow Graph Feature Matching*

Berikut ini adalah *flow graph Feature Matching*:



Gambar 4.4 *Flow Graph Feature Matching*

Keterangan :

◯ = Menggambarkan Kondisi

→ = Menggambarkan Aksi

b. *Graph Matriks Feature Matching*

Berikut ini adalah *graph matriks* dari *flow graph feature matching :*

Tabel 4-8 *Graph Matriks Feature Matching*

	1	2	3	4	5	6	7	8	9	10
1		1								1
2			1							
3				1			1			
4					1			1		
5						1			1	
6			1				1			
7				1				1		
8					1				1	
9	1									
10										

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

5. Pengujian *Eye Tracking*

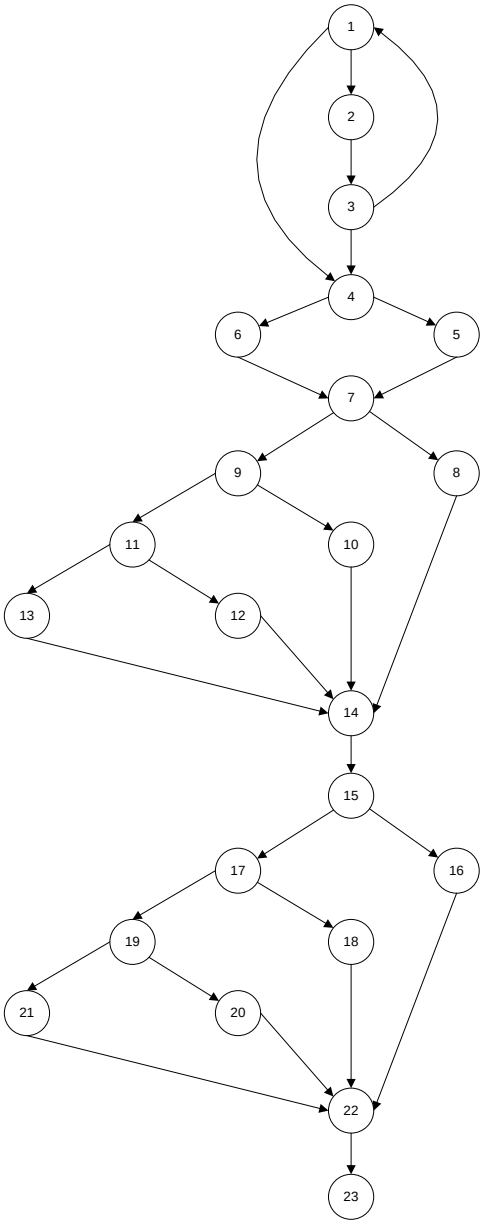
Berikut ini adalah algoritma eye tracking yang akan diuji :

```
1.    for (int i = 0; i < eyes.Total; i++)
2.    {
       CvRect r = eyes[0].Value.Rect;
       CvPoint center = new CvPoint
       {
         X = Cv.Round((r.X + r.Width * 0.5) * Scale),
         Y = Cv.Round((r.Y + r.Height * 0.5) * Scale)
3.     };
      img.DrawRect(r, CvColor.Red);
      int radius = Cv.Round((r.Width + r.Height) * 0.15 * Scale);
      img.DrawRect(r, colors[0 % 8]);
      img.Rectangle(r,colors[0 % 8], 3, LineType.AntiAlias,0);
      img.Rectangle(Convert.ToInt32((r.X + r.Width * 0.5) *
      Scale),Convert.ToInt32((r.Y + r.Height * 0.5) * Scale),
```

```
      Convert.ToInt32(r.Width * Scale), Convert.ToInt32(r.Height *
      Scale),colors[0 % 8], 2, LineType.AntiAlias, 0);
4.     if (sett == true)
5.     {
        xt = center.X;
        yt = center.Y;
        sett = false;
6.     }
       Xok = 0;
7.     if (Math.Abs(center.X - xt) >= 10 && Math.Abs(center.X -
       xt) <=100)
8.     {
9.      if (center.X > xt)
10.     {
         Xok = 10;
        }
11.     Else if (center.X < xt)
12.     {
        Xok = -10;
        }
13.     Else
        Xsblm = center.X;
14.    }
       Yok = 0;
15.    if (Math.Abs(center.Y - yt) >= 10 && Math.Abs(center.Y -
       yt) <=  100)
16.    {
17.     if (center.Y > yt)
18.     {
         Yok = 10;
        }
19.     Else if (center.Y < yt)
20.     {
         Yok = -10;
        }
21.     Else Ysblm = center.Y;
22.    }
23.    Cursor.Position = new Point
      (Cursor.Position.X + Xok, Cursor.Position.Y + Yok);
```

a. *Flow Graph Eye Tracking*

Berikut ini adalah *flow graph Eye Tracking*:



Gambar 4.5 *Flow Graph Eye Tracking*

Keterangan :

◯ = Menggambarkan Kondisi

→ = Menggambarkan Aksi

b. *Graph Matriks Eye Tracking*

Berikut ini adalah *graph matriks* dari *flow graph eye tracking :*

Tabel 4-9 *Graph Matriks Eye Tracking*

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

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

4.2.1.2 Pengujian *Beta*

Pengujian *beta* dilakukan dengan pengujian performansi yaitu pengujian uji coba langsung kepada pengguna sistem untuk mencoba aplikasi yang baru dan hasil dari percobaan tersebut dicatat untuk mengetahui tingkat keakuratan sistem. Berikut ini adalah hasil dari pengujian performasi pada perangkat lunak pengendali pointer :

1. Pengujian performansi pendeteksian mata

Pengujian pendeteksian mata dilakukan pada 3 orang pengguna dengan masing–masing melakukan 10 kali pendeteksian pada ruangan yang pencahayaannya cukup dan pada ruangan yang pencahayaannya kurang. Berikut ini adalah hasil dari pengujian untuk mendeteksi mata pada ruangan yang pencahayaannya cukup:

Tabel 4-10 Hasil Pengujian Pendeteksian mata pada ruangan cukup cahaya

No	Nama	Keterangan	
		Terdeteksi	Tidak terdeteksi
1	Anna (Mata Sipit)	7	3
2	Jessy (Menggunakan kacamata)	7	3
3	Agus (Mata Besar)	8	2
Tingkat keakuratan		$\frac{7+7+8}{30}$ x $100 = 73{,}3\ \%$	

Berikut ini adalah hasil dari pengujian untuk mendeteksi mata pada ruangan yang pencahayaannya kurang:

Tabel 4-11 Hasil Pengujian Pendeteksian mata pada ruangan kurang cahaya

No	Nama	Keterangan	
		Terdeteksi	Tidak terdeteksi
1	Anna (Mata Sipit)	2	8
2	Jessy (Menggunakan kacamata)	0	10
3	Agus (Mata Besar)	1	9
Tingkat keakuratan		$\frac{2+0+1}{30}$ x 100 = 3 %	

2. Pengujian performansi perintah pada pointer

Pengujian perintah pada pointer dilakukan pada 3 orang pengguna dan masing-masing pengguna melakukan 3 perintah dan setiap perintah dilakukan 5 kali percobaan. Berikut ini adalah hasil dari pengujian untuk perintah :

Tabel 4-12 Hasil Pengujian Perintah

No	Nama	Perintah	Waktu (detik)	Keterangan
1	Anna (Mata Sipit)	Pergerakan pointer	4	Tepat
		Pergerakan pointer	3	Tepat
		Pergerakan pointer	4	Tepat
		Pergerakan pointer	4	Tepat
		Pergerakan pointer	5	Tepat
		Klik kanan	2	Tepat
		Klik kanan	2	Tepat
		Klik kanan	3	Tepat

No	Nama	Perintah	Waktu (detik)	Keterangan
		Klik kanan	3	Tepat
		Klik kanan	3	Tepat
		Klik Kiri	1	Tepat
		Klik Kiri	1	Tepat
		Klik Kiri	2	Tepat
		Klik Kiri	2	Tepat
		Klik Kiri	1	Tepat
2	Jessy (Menggunakan kacamata)	Pergerakan pointer	4	Tepat
		Pergerakan pointer	5	Tepat
		Pergerakan pointer	4	Tepat
		Pergerakan pointer	4	Tepat
		Pergerakan pointer	5	Tepat
		Klik kanan	5	Tepat
		Klik kanan	4	Tepat
		Klik kanan	5	Tepat
		Klik kanan	3	Tepat
		Klik kanan	2	Tepat
		Klik Kiri	2	Tepat
		Klik Kiri	3	Tepat
		Klik Kiri	2	Tepat
		Klik Kiri	3	Tepat
		Klik Kiri	1	Tepat
	Agus (Mata Besar)	Pergerakan pointer	3	Tepat
		Pergerakan pointer	3	Tepat
		Pergerakan pointer	2	Tepat
		Pergerakan pointer	2	Tepat
		Pergerakan pointer	5	Tepat
		Klik kanan	2	Tepat
		Klik kanan	1	Tepat
		Klik kanan	2	Tepat
		Klik kanan	2	Tepat
		Klik kanan	3	Tepat
		Klik Kiri	2	Tepat
		Klik Kiri	1	Tepat
		Klik Kiri	1	Tepat
		Klik Kiri	2	Tepat
		Klik Kiri	2	Tepat
Waktu rata – rata untuk melakukan perintah			123 / 45 = 2,7 detik	

3. Kesimpulan Pengujian *Beta*

Berdasarkan hasil pengujian performansi diatas, maka diambil kesimpulan bahwa perangkat lunak pengendali pointer ini memiliki tingkat keakuratan sekitar 70 % dan waktu eksekusi perintah sekitar 0-3 detik.