BAB IV
PERANCANGAN DAN IMPLEMENTASI

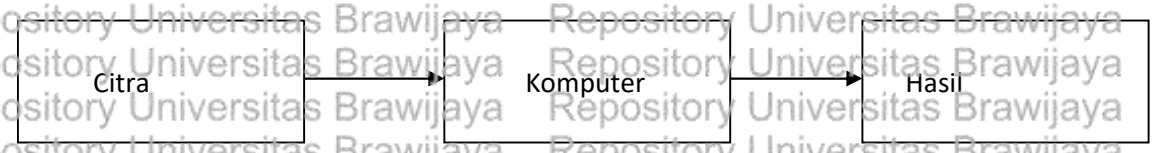
Bab ini menjelaskan mengenai langkah-langkah yang akan dilakukan untuk merancang dan mengimplementasikan aplikasi identifikasi berbagai jenis objek yang tercampur berdasarkan warna dan bentuk. Perancangan dan implementasi dikerjakan dengan beberapa tahap meliputi basis data, pengambilan gambar, *preprocessing*, ekstraksi ciri, pembelajaran LVQ, segmentasi dan identifikasi objek. Perancangan diawali dengan penggambaran blok diagram kerja sistem yang menunjukkan cara kerja aplikasi secara umum.

4.1 Perancangan Secara Umum

Perancangan aplikasi secara umum merupakan tahap awal dalam melakukan perancangan aplikasi identifikasi objek yang akan dibuat. Perancangan diawali dengan penggambaran blok diagram kerja sistem yang menunjukkan cara kerja aplikasi secara umum.

4.1.1 Blok diagram sistem

Pembuatan diagram sistem merupakan dasar dari perancangan sistem agar perancangan dan perealisasian aplikasi berjalan secara sistematis. Citra masukan adalah citra objek yang ditangkap menggunakan kamera. Blok diagram dapat ditunjukkan seperti pada gambar 4.1.



Gambar 4.1 Blok Diagram
(**Sumber**: Perancangan)

Fungsi dari masing-masing bagian pada diagram blok ini akan dijelaskan sebagai berikut:

1. Citra objek merupakan gambar yang merepresentasikan objek yang akan digunakan sebagai input sistem.

2. Komputer digunakan sebagai sarana untuk proses pengolahan citra mulai dari pre proses, *image processing*, pelatihan dan untuk mendeteksi sekaligus menghitung objek.

3. Dari proses dua tahap sebelumnya maka akan didapatkan informasi yang menampilkan hasil identifikasi yang sesuai dengan tujuan perancangan dan pembuatan aplikasi ini.

Gambar 4.2 Diagram proses

(**Sumber:** Perancangan)

Gambar 4.2 menunjukkan gambaran diagram proses secara keseluruhan. Aplikasi berjalan dalam 2 fase yaitu fase pelatihan dan fase pengujian.

Pada fase pelatihan, gambar objek masukan merupakan gambar objek kacang individu atau objek kacang tunggal, kemudian citra objek dilakukan *preprocessing* dahulu sebelum di ekstraksi ciri warna dan bentuknya. Ciri warna yang diambil yaitu nilai rata-rata intensitas *red, green, blue* dan untuk ciri bentuk yaitu diambil nilai *roundness*.

Setelah ciri didapat dan disimpan dalam database selanjutnya masuk ke proses *learning* (pembelajaran), yang kemudian satu persatu ciri kacang tersebut dilatih menggunakan pembelajaran LVQ. Pembelajaran LVQ merupakan salah satu metode pengklasifikasian pada jaringan syaraf tiruan. Dari hasil pembelajaran tersebut maka akan didapat nilai bobot yang nantinya akan dijadikan acuan untuk mengidentifikasi objek dengan cara menghitung jarak antara bobot yang telah tersimpan dengan ekstraksi ciri dari objek uji menggunakan metode *euclidean distance*, proses ini berjalan pada fase pengujian. Dari proses tersebut akan diketahui jenis dan jumlah kacang dari tiap-tiap jenis yang ada pada citra input dan informasi yang didapat kemudian di-*output*kan.

4.1.2 Cara kerja aplikasi

Cara kerja aplikasi identifikasi dan penghitung berbagai objek yang tercampur berdasarkan warna dan bentuk dimulai dari fase pelatihan dengan proses me-*load* citra yang didapat dari pengambilan gambar melalui kamera. Kemudian dilanjutkan dengan proses *resize* dan merubah citra menjadi citra *bitmap* agar dapat diproses dengan baik menggunakan teknik pengolahan citra.

Setelah melalui tahap *preprocessing* yang bertujuan agar objek kacang terpisahkan dengan latar belakangnya, maka dilakukan ekstraksi ciri warna dan bentuk. Ciri warna yang diambil adalah nilai rata-rata intensitas piksel kanal RGB sedangkan ciri bentuk adalah nilai *roundness* yang didapat berdasarkan nilai luas piksel objek dan keliling pixel objek. Hasil ekstraksi ciri akan disimpan dalam *database* untuk proses pelatihan LVQ.

Dalam proses pelatihan LVQ dibutuhkan suatu masukan vektor yaitu vektor ciri hasil proses ektraksi. Setiap masukan memiliki vektor ciri yang direpresentasikan dalam matriks berukuran 1x4. Selanjutnya *user* dapat mengatur nilai parameter *learning rate,* dan *maximum epoch*. Dari pelatihan tersebut akan didapatkan nilai bobot yang kemudian disimpan dalam database dan akan digunakan sebagai acuan identifikasi.

Pada fase pengujian sendiri diawali dengan proses *preprocessing* yang memiliki tujuan sama yaitu memisahkan objek dengan latar belakang, pada tahap selanjutnya dikarenakan objek yang terdapat pada citra adalah objek jamak maka perlu dilakukan proses *segmentasi* terhadap citra masukan dengan tujuan memisahkan tiap-tiap objek dengan objek yang lain, kemudian tiap –tiap objek tersebut akan diidentifikasi dengan cara diekstraksi ciri, setelah didapat hasil ekstrasi ciri kemudian dari tiap objek tersebut dilakukan penghitungan jarak dengan menghitung *euclidean distance* antara bobot tiap

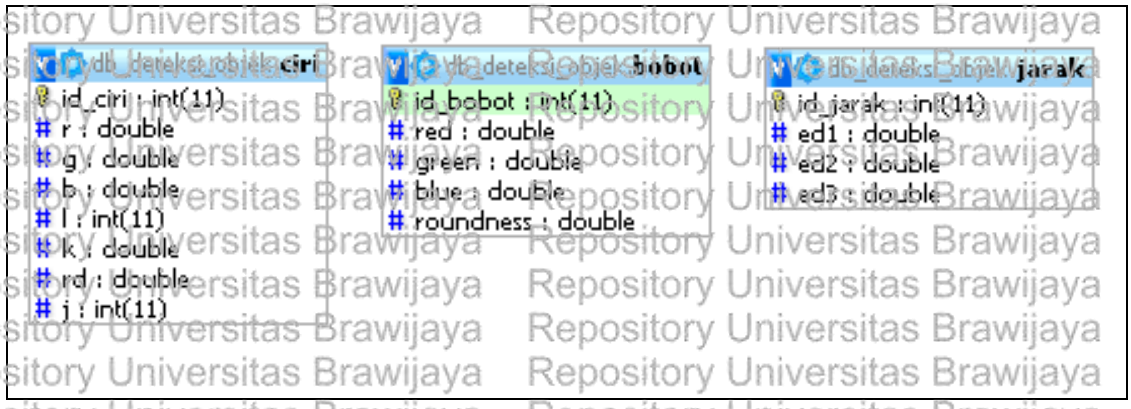
jenis objek latih dengan objek pengujian. Jika didapatkan jarak terdekat terhadap salah satu jenis objek tertentu maka objek uji tersebut akan masuk ke dalam jenis objek yang sama dengan objek yang telah ditentukan tersebut, proses ini dilakukan hingga seluruh objek dalam citra telah diidentifikasi.

4.2 Perancangan Perangkat Lunak

Pada sub bab perancangan perangkat lunak akan membahas secara detail tentang setiap tahapan proses pada aplikasi identifikasi dan penghitung berbagai jenis objek yang tercampur berdasarkan warna dan bentuk. Beberapa tahap perancangan tersebut meliputi basis data, *load* gambar, pemisahan *background*, ekstraksi ciri, pembelajaran *Learning Vector Quantization(LVQ)*, segmentasi, identifikasi objek. Perancangan perangkat lunak diimplementasikan dalam bahasa pemrograman C#.

4.2.1 Basis data

Perancangan basis data dibangun menggunakan MySQL. Basis data ini akan digunakan sebagai tempat penyimpanan hasil ekstraksi ciri dan juga bobot. Beberapa tabel yang dibutuhkan adalah tabel ciri, bobot, dan jarak seperti yang ditunjukkan pada gambar 4.3.



Gambar 4.3 Tabel basis data

(**Sumber**: Perancangan)

Gambar 4.3 menunjukkan bahwa tabel ciri akan menyimpan nilai hasil ekstraksi ciri seperti *red*(r), *green*(g), *blue*(b), *luas*(l), *keliling*(k), *roundness*(rd). Selain itu tabel ciri juga akan menyimpan beberapa informasi penting seperti jenis objek(j). Tabel bobot menyimpan bobot hasil pelatihan LVQ untuk setiap jenis objek yaitu nilai *red*, *green*, *blue*, dan *roundness*. Tabel jarak menyimpan nilai jarak setiap pelatihan

LVQ dimana terdapat jarak terhadap kelas 1 sampai kelas 3 yang diwakili dengan ed1, ed2, dan ed3.

Menghubungkan basis data dengan perangkat lunak pada bahasa C# menggunakan MySQL *Connector*. Berikut ini adalah bagian kode bahasa C# agar dapat terhubung dengan basis data:

```
string connect_db =
"server=localhost;user=root;database=db_deteksi_objek;port=3306;password=";
MySqlConnection connect = new MySqlConnection(connect_db);
```

4.2.2 Load Image

Sebelum memulai proses pengolahan citra aplikasi ini membutuhkan *input* berupa gambar yang merepresentasikan citra objek, sebelumnya citra objek ini didapatkan dengan cara pengambilan gambar melalui kamera yang dilakukan diluar proses aplikasi berlangsung dengan kata lain pengambilan gambar melalui kamera tidak masuk dalam urutan proses saat aplikasi berjalan dan tidak dibahas dalam penulisan ini.

Citra yang didapat dari pengambilan gambar memiliki beragam ukuran, agar semua citra memiliki keseragaman ukuran maka sebelum ditampilkan citra objek tersebut akan melalui tahap *resize* dengan ukuran 320x240 piksel dan diberi format bitmap (BMP).

Berikut ini adalah bagian kode bahasa C# yang berfungsi mengaktifkan menu dialog untuk memilih sumber gambar :

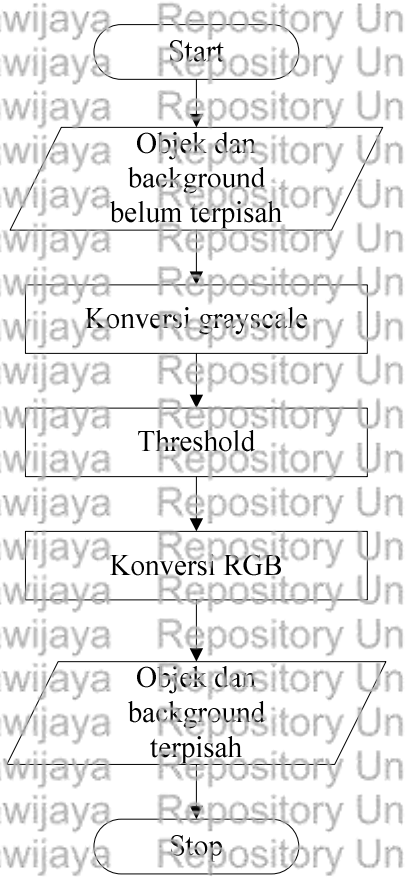
```
OpenFileDialog menu = new OpenFileDialog();
DialogResult result = menu.ShowDialog();
```

Untuk resize dan konversi format sebagai berikut :

```
Image gambar = Image.FromFile(menu.FileName);
Bitmap tampil = new Bitmap(gambar, 320, 240);
```

4.2.3 Preprocessing

Setelah melalui tahap *load image* dan citra objek telah siap untuk digunakan maka sebelum proses berjalan lebih jauh citra objek akan melalui tahap *preprocessing*. *Preprocessing* yang dilakukan bertujuan memisahkan citra objek dengan latar belakangnya. Proses *preprocessing* yang berlangsung yaitu konversi *grayscale, thresholding* dan konversi RGB seperti pada gambar 4.4.

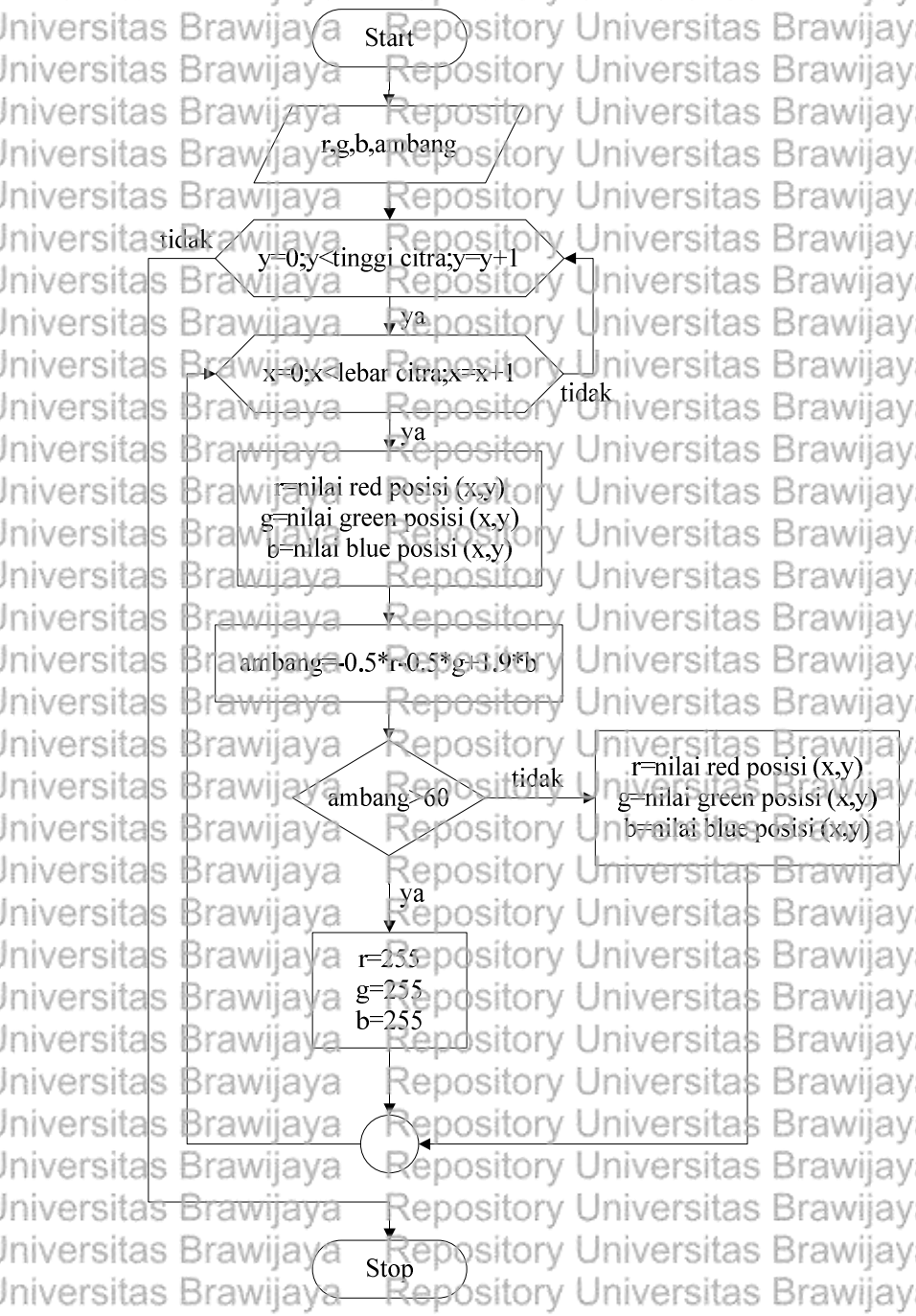


Gambar 4.4 Flowchart *preprocessing*

(**Sumber:** perancangan)

Gambar 4.4 menjelaskan bahwa citra yang menjadi masukan akan dilakukan konversi ke dalam bentuk *grayscale* karena hasil pada tahap *load image* berupa citra warna atau RGB. Tujuan konversi *grayscale* ini adalah supaya kombinasi warna menjadi lebih sederhana sehingga memudahkan untuk proses selanjutnya. Perhitungan *grayscale* yaitu $-0.5 * R - 0.5 * G +$. *Grayscale* menyebabkan citra warna menjadi citra abu-abu yang memiliki rentang nilai piksel mulai 0 – 255.

Selanjutnya adalah melakukan *thresholding* atau pengambangan. Proses *thresholding* bertujuan untuk menghilangkan latar belakang citra. Nilai *threshold* ideal pada proses ini adalah 60, yang didapatkan dengan melakukan *trial* dan *error*. Perhitungan *threshold* adalah dengan mengubah piksel bernilai >60 menjadi berwarna putih (255, 255, 255). Sedangkan piksel bernilai ≤60 akan dikembalikan menjadi warna aslinya.



Gambar 4.5 Flowchart pemisahan *background*

(**Sumber:** Perancangan)

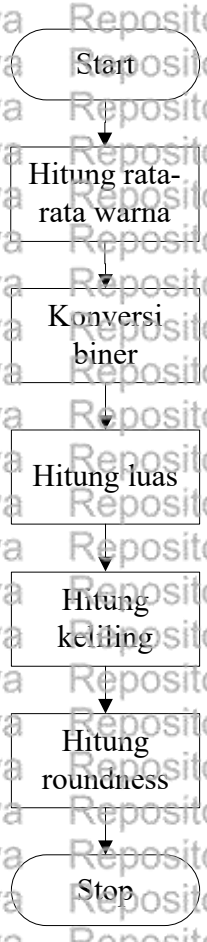
Implementasi proses *preprocessing* pada bahasa C# adalah sebagai berikut:

```
gambar_pisah = (Bitmap)pictureBox1.Image.Clone();
for (int y = 0; y < gambar_pisah.Height; y++)
{
 for (int x = 0; x < gambar_pisah.Width; x++)
 {
```

```
int r = gambar_pisah.GetPixel(x, y).R;
int g = gambar_pisah.GetPixel(x, y).G;
int b = gambar_pisah.GetPixel(x, y).B;
int ambang = -0.5*r – 0.5 * g + 1.9 * b;
if (ambang > 60)
{
        r = 255;
        g = 255;
        b = 255;
}
gambar_pisah.SetPixel(x, y, Color.FromArgb(r, g, b));
}
}
pictureBox1.Image = gambar_pisah;
```

4.2.4 Perancangan ekstraksi ciri

Pada tahap ekstraksi ciri citra masukan berupa citra hasil pemisahan *background*. Ekstraksi ciri bertujuan untuk mendapatkan karakteristik dari suatu objek tertentu yang dapat membedakan dengan jenis objek yang lain. Ciri yang digunakan adalah ciri warna dan ciri bentuk. Ciri warna yang diambil adalah rata-rata (*mean*) warna RGB, sedangkan ciri bentuk adalah nilai *roundness*. Untuk mendapatkan ciri warna dan bentuk digunakan beberapa metode seperti yang dijelaskan pada gambar 4.6.



Gambar 4.6 Flowchart ekstraksi ciri

(**Sumber:** Perancangan)

Gambar 4.6 menunjukkan bahwa citra hasil pemisahan *backgound* merupakan masukan dalam tahap ekstraksi ciri. Setelah mendapat objek yang diinginkan maka dilakukan perhitungan *mean* warna. Selanjutnya citra dikonversi ke dalam bentuk biner untuk memudahkan dalam proses selanjutnya. Apabila perhitungan luas (*area*) selesai maka dilakukan operasi morfologi guna mendapatkan tepian citra yang berfungsi untuk memperoleh keliling (*perimeter*) sehingga akan diperoleh nilai *roundness*.

4.2.4.1 Ekstraksi Ciri Warna

Metode yang digunakan untuk mendapatkan ciri berdasarkan warna adalah dengan menggunakan *mean* warna. *Mean* warna adalah nilai rata-rata kanal dari suatu objek. *Mean* dihitung pada setiap piksel untuk masing-masing nilai *kanal* piksel. Ekstraksi ciri warna didapat dengan menghitung intensitas rata-rata pixel kanal merah, hijau dan biru objek dengan rumus sebagai berikut:

$$r_{avg} = \frac{\sum_{i=1}^{N} r(i)}{N}; \; g_{avg} = \frac{\sum_{i=1}^{N} g(i)}{N}; \; b_{avg} = \frac{\sum}{}$$

dengan:

r_{avg}, g_{avg}, b_{avg} = rata-rata intensitas warna R, G, B

r(i), g(i), b(i) = nilai intensitas warna R, G, B pada suatu pixel

N = total pixel pada objek.

Ekstraksi ciri warna dilakukan dengan cara mengambil setiap nilai RGB pixel pada objek. Pengambilan nilai dengan cara memeriksa pixel yang tidak berwarna putih atau bernilai R=255, G=255, B=255. Selanjutnya nilai RGB disimpan dalam suatu variabel yang nantinya akan dijumlahkan dengan nilai RGB pada iterasi selanjutnya. Proses ini berlangsung selama belum mencapai pada koordinat pixel terakhir. Setelah perulangan berakhir maka dilakukan perhitungan nilai rata-rata RGB yang didapat dengan cara membagi jumlah nilai RGB dari perulangan dengan jumlah pixel objek pada citra. Penjelasan terdapat pada Gambar 4.7 dibawah ini.

Gambar 4.7 Flowchart ekstraksi ciri warna

(Sumber: Perancangan)

Implementasi proses ekstraksi ciri warna pada bahasa C# adalah sebagai berikut:

```
gambar_ekstraksi = (Bitmap)pictureBox2.Image.Clone();
jumlah = 0;

for (int y = 0; y < gambar_ekstraksi.Height; y++)
{
  for (int x = 0; x < gambar_ekstraksi.Width; x++)
  {
    Red = gambar_ekstraksi.GetPixel(x, y).R;
    Green = gambar_ekstraksi.GetPixel(x, y).G;
    Blue = gambar_ekstraksi.GetPixel(x, y).B;

    if (Red != 255 && Green != 255 && Blue != 255)
    {
      Rrata += Red;
      Grata += Green;
      Brata += Blue;
      jumlah = jumlah + 1;
    }
  }
}
Rrata = Rrata / jumlah;
Grata = Grata / jumlah;
Brata = Brata / jumlah;
```

4.2.4.2 Ekstraksi Ciri Bentuk

Untuk ciri bentuk yang nilainya akan diambil dan diproses dari objek adalah nilai kebundaran (*roundness*). Nilai kebundaran didapat dengan mencari terlebih dahulu nilai luas (*area*) dan keliling (*perimeter*) objek. Ekstraksi ciri bentuk dimulai dengan merubah citra warna ke *grayscale* kemudian merubahnya lagi ke dalam bentuk biner. Selanjutnya menghitung luas dan keliling objek.

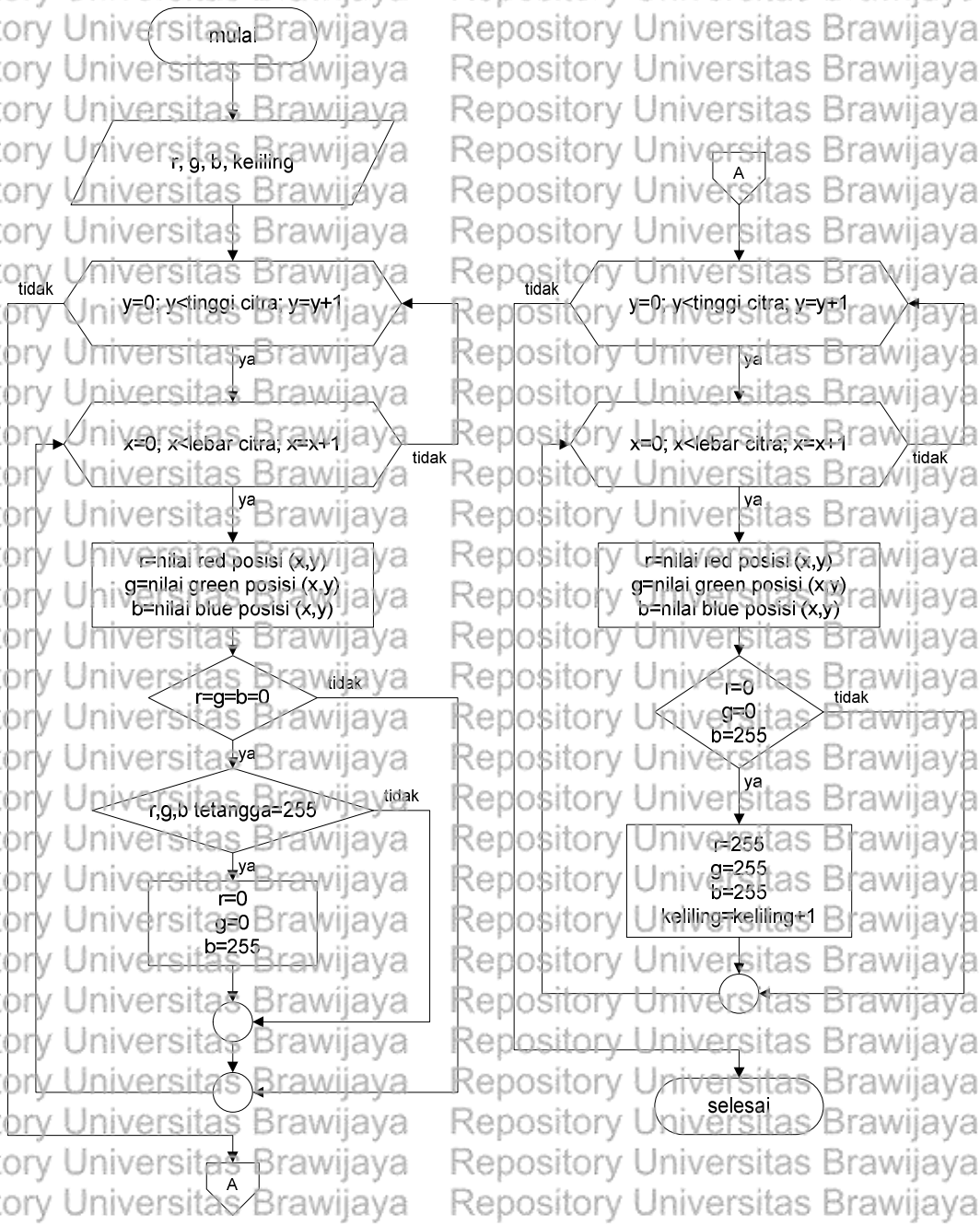
Nilai luas dan keliling diestimasikan dalam pixel. Keliling (*perimeter*) didapat dengan cara menghitung jumlah pixel yang berbatasan dengan *background*, sedangkan untuk luas (*area*) didapat dengan cara menghitung jumlah pixel yang tertutup oleh keliling objek. Untuk mendapatkan nilai keliling (*perimeter*) digunakan metode yang hampir sama dengan operasi morfologi, dimana operasi tersebut akan melakukan proses erosi terhadap objek akan tetapi tidak sampai melakukan pengurangan citra, dan hanya sekedar mendeteksi tepian objek saja. Untuk proses mendapakan luas objek dapat dilihat pada Gambar 4.8 dibawah ini.

Gambar 4.8 Flowchart mencari nilai luas

(**Sumber:** perancangan)

Pada Gambar 4.8 terdapat urutan proses ekstraksi bentuk untuk mendapatkan luas (*area*) objek. Luas objek yang dimaksud adalah jumlah piksel atau keseluruhan piksel yang dinyatakan sebagai milik objek tersebut, maka dari itu dilakukan

pengecekan mana saja piksel yang direpresentasikan sebagai piksel objek. Pada proses sebelumnya telah dilakukan pemisahan antara objek dengan background, dimana piksel background diubah menjadi warna putih (R=G=B=255), dari sini dilakukan pengecekan terhadap mana saja piksel yang tidak berwarna putih dan piksel yang tidak putih tersebut diubah menjadi warna hitam (R=G=B=0). Jumlah nilai piksel berwarna hitam inilah yang merupakan luas dari objek.



Gambar 4.9 Flowchart mencari nilai keliling

(**Sumber:** perancangan)

Gambar 4.9 merupakan tahapan proses ekstraksi bentuk untuk mendapatkan keliling (*perimeter*) objek menggunakan operasi morfologi. Operasi ini bertujuan untuk mencari piksel yang berbatasan antara background dengan objek. Proses ini dimulai dengan memberikan inisialisasi terhadap 8 arah hubungan piksel. Selanjutnya dilakukan pengecekan terhadap piksel objek, apabila piksel objek bersebelahan dengan piksel background maka piksel tersebut diubah menjadi biru(r=0,g=0,b=255). Proses selanjutnya yaitu mendapatkan nilai keliling dengan cara menghitung jumlah piksel yang berwarna biru tersebut, sehingga nilai *roundness* objek dapat dicari menggunakan rumus:

$$C = 4\pi \frac{Area}{Perimeter^2}$$

dengan:

C = kebundaran objek
Area = luas objek
Perimeter = keliling objek

Gambar 4.10 Flowchart menghitung roundness

(**Sumber:** perancangan)

Implementasi proses ekstraksi bentuk pada bahasa C# adalah sebagai berikut:

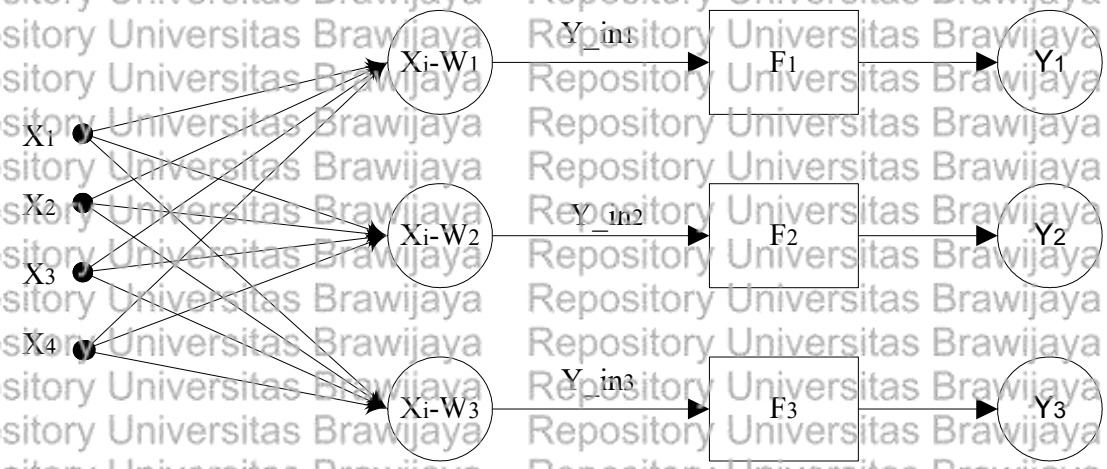
```
// - Mencari Luas
luas = 0;
for (int y = 0; y < gambar_luas.Height; y++)
{
    for (int x = 0; x < gambar_luas.Width; x++)
    {
        int r = gambar_luas.GetPixel(x, y).R;
        int g = gambar_luas.GetPixel(x, y).G;
        int b = gambar_luas.GetPixel(x, y).B;

        if (r != 255 && g != 255 && b != 255)
        {
            luas = luas + 1;
            r = 0;
            g = 0;
            b = 0;
        }
        gambar_luas.SetPixel(x, y, Color.FromArgb(r, g, b));
    }
}
pictureBox4.Image = gambar_luas;

//    Mencari Tepi Dan Keliling
keliling = 0;
for (int y = 0; y < gambar_tepi.Height; y++)
{
    for (int x = 0; x < gambar_tepi.Width; x++)
    {
        int r = gambar_tepi.GetPixel(x, y).R;
        int g = gambar_tepi.GetPixel(x, y).G;
        int b = gambar_tepi.GetPixel(x, y).B;
        if (r == 0 && g == 0 && b == 0)
        {
            //kanan
            int rka = gambar_tepi.GetPixel(x + 1, y).R;
            int gka = gambar_tepi.GetPixel(x + 1, y).G;
            int bka = gambar_tepi.GetPixel(x + 1, y).B;
            //kanan bawah
            int rkab = gambar_tepi.GetPixel(x + 1, y + 1).R;
            int gkab = gambar_tepi.GetPixel(x + 1, y + 1).G;
            int bkab = gambar_tepi.GetPixel(x + 1, y + 1).B;
            //bawah
            int rb = gambar_tepi.GetPixel(x, y + 1).R;
            int gb = gambar_tepi.GetPixel(x, y + 1).G;
            int bb = gambar_tepi.GetPixel(x, y + 1).B;
            //kiri bawah
            int rkib = gambar_tepi.GetPixel(x - 1, y + 1).R;
            int gkib = gambar_tepi.GetPixel(x - 1, y + 1).G;
            int bkib = gambar_tepi.GetPixel(x - 1, y + 1).B;
            //kiri
            int rki = gambar_tepi.GetPixel(x - 1, y).R;
            int gki = gambar_tepi.GetPixel(x - 1, y).G;
            int bki = gambar_tepi.GetPixel(x - 1, y).B;
            //kiri atas
            int rkia = gambar_tepi.GetPixel(x - 1, y - 1).R;
            int gkia = gambar_tepi.GetPixel(x - 1, y - 1).G;
            int bkia = gambar_tepi.GetPixel(x - 1, y - 1).B;
            //atas
            int ra = gambar_tepi.GetPixel(x, y - 1).R;
            int ga = gambar_tepi.GetPixel(x, y - 1).G;
            int ba = gambar_tepi.GetPixel(x, y - 1).B;
```

```
            //kanan atas
            int rkaa = gambar_tepi.GetPixel(x + 1, y - 1).R;
            int gkaa = gambar_tepi.GetPixel(x + 1, y - 1).G;
            int bkaa = gambar_tepi.GetPixel(x + 1, y - 1).B;
            if ((ra == 255 && ga == 255 && ba == 255) || (rki == 255 && gki == 255 &&
            bki == 255) || (rka == 255 && gka == 255 && bka == 255) || (rb == 255 &&
              gb == 255 && bb == 255) || (rkia == 255 && gkia == 255 && bkia == 255)
              || (rkaa == 255 && gkaa == 255 && bkaa == 255) || (rkib == 255 && gkib
              == 255 && bkib == 255) || (rkab == 255 && gkab == 255 && bkab == 255))
            {
              r = 0;
              g = 0;
              b = 255;
            }
          }
          gambar_tepi.SetPixel(x, y, Color.FromArgb(r, g, b));
        }
      }
      for (int y = 0; y < gambar_tepi.Height; y++)
      {
        for (int x = 0; x < gambar_tepi.Width; x++)
        {
          int r = gambar_tepi.GetPixel(x, y).R;
          int g = gambar_tepi.GetPixel(x, y).G;
          int b = gambar_tepi.GetPixel(x, y).B;
          if (r == 255 && g == 255 && b == 255)
          {
            r = 0;
            g = 0;
            b = 0;
          }
          if (r == 0 && g == 0 && b == 255)
          {
            keliling = keliling + 1;
            r = 255;
            g = 255;
            b = 255;
          }
          gambar_tepi.SetPixel(x, y, Color.FromArgb(r, g, b));
        }
      }
      //Mencari Nilai Roundness
      roundness = (double)((4 * Math.PI * luas) / (keliling * keliling));
```

Sebanyak 60 objek pelatihan mengalami ekstraksi ciri secara bergantian, dan secara acak. Hasil dari ekstraksi ciri ini nilai-nilainya akan disimpan ke dalam basis data.

4.2.5 Perancangan pembelajaran *Learning Vector Quantization*

Proses pembelajaran LVQ yang dilakukan pertama kali adalah pengambilan data training. Data training ini diambil dari basis data yang berisi nilai-nilai ekstraksi ciri. Data training ini merupakan input *layer* pada jaringan saraf tiruan seperti yang ditunjukkan pada gambar 4.11.



Gambar 4.11 Struktur jaringan LVQ

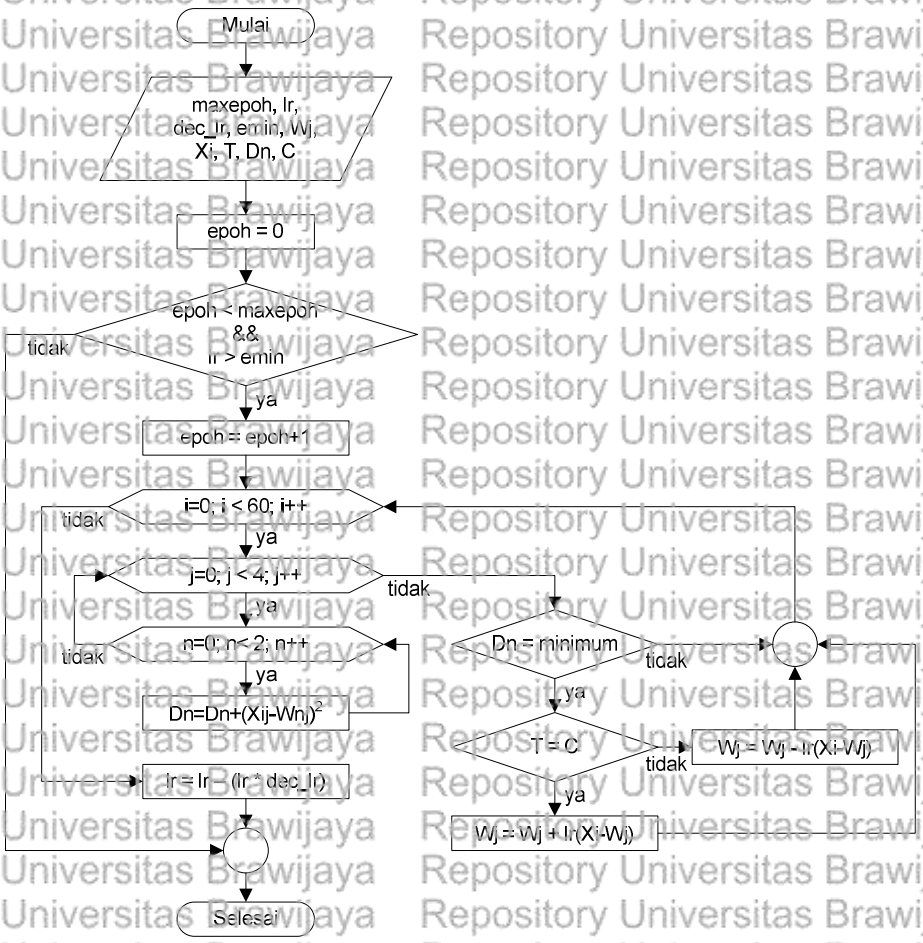
(**Sumber:** Perancangan)

Gambar 4.11 adalah arsitektur jaringan LVQ (*Learning Vector Quantization*) pada aplikasi yang terdiri dari 4 *node* lapisan masukan (*input layer*), 3 *node* lapisan tersembunyi (*hidden layer*) serta 3 *node* lapisan keluaran (*output layer*). Lapisan input memiliki 4 node yang disimbolkan dengan nilai x_1, x_2, x_3, dan x_4. Pada tiap lapisan masukan terlebih dahulu diberikan 3 nilai bobot yang berbeda yaitu W_1, W_2, dan W_3. Pemberian bobot bertujuan agar nilai tiap masukan memiliki penimbang (bobot) yang kemudian akan dihitung nilainya dan diselesaikan dengan persamaan yang dimiliki oleh metode LVQ (*Learning Vector Quantization*), sehingga jaringan memiliki 3 kelas yang berbeda, yaitu kelas Y_1, Y_2, Y_3. Kelas Y_1 memiliki bobot W_1, kelas Y_2 memiliki bobot W_2 dan seterusnya. Keluaran dari lapisan ini akan menjadi masukan bagi lapisan tersembunyi (*hidden layer*) sebanyak 3 *node* yaitu Y_in_1, Y_in_2, dan Y_in_3.

Bobot awal pada aplikasi ini diberikan dengan mengambil nilai data pelatihan pertama untuk setiap kelasnya. Dengan menggunakan metode *euclidean distance* bahwa nilai paling kecil yang dihasilkan adalah pemenang dan merupakan kelas dari input tersebut maka pada lapisan keluaran (*output layer*) digunakan sebuah fungsi pembanding yang berguna membandingkan dua nilai tersebut untuk dicari nilai terkecilnya. Dalam gambar 4.11 fungsi pembanding tersebut dituliskan dengan simbol

F_1, F_2, dan F_3. Pada aplikasi ini nilai *learning rate* ditentukan oleh user begitu juga nilai pengurangan *learning rate*.

Nilai input yang diberikan ke lapisan masukan (input layer) berupa vektor. Tujuannya adalah untuk mempermudah penentuan jumlah node pada lapisan masukan serta perhitungannya dengan metode LVQ (Learning Vector Quantization).



Gambar 4.12 Flowchart algoritma LVQ

(**Sumber:** perancangan)

Gambar 4.12 menjelaskan algoritma jaringan saraf tiruan metode LVQ. Langkah pertama adalah menentukan masing-masing kelas output (C), bobot (W), *learning rate* (lr), maksimum epoh (maxepoh), dan eror minimum yang diharapkan (emin). Kemudian memasukkan data input (X_i) dengan i=1, 2, 3, ..., 60 serta target kelas (T).

Masing-masing input dibandingkan dengan bobot yang telah ditetapkan dengan melakukan pengukuran jarak antara masing-masing bobot W dan input X. Nilai minimum dari hasil perbandingan akan menentukan kelas dari vektor input dan

perubahan bobot dari kelas tersebut. Untuk input dan bobot yang memiliki kelas yang sama maka bobot diubah menjadi $W = W + Lr$ sedangkan untuk input dan bobot dengan kelas yang berbeda diubah menjadi $W = W - Lr$. Proses ini dilakukan secara terus-menerus sampai kondisi maxepoh dan learning rate terpenuhi.

Berikut ini adalah salah satu contoh pengukuran dan pembandingan jarak pada aplikasi untuk 3 input pertama.

Vektor input objek:

- Kacang Merah = [107.96 46.09 26.06 0.59]
- Kacang Merah = [113.67 49.59 28.19 0.60]
- Kacang Hijau = [70.20 61.53 23.78 0.69]

Vektor bobot masing-masing jenis objek:

- Kacang Merah = [91.28 38.05 26.00 0.58]
- Kacang Hijau = [72.28 61.19 23.08 0.72]
- Kacang Kedelai = [141.39 106.32 69.15 0.66]

Proses pelatihan yang terjadi adalah sebagai berikut:

Iterasi 1:

Data ke-1 [107.96 46.09 26.06 0.59] mewakili kacang merah:

- Jarak ke bobot 1 (jenis objek kacang merah)

$$= \sqrt{(107.96 - 91.28)^2 + (46.09 - 38.05)^2 + (26.06 - 26.00)^2 + (0.59}$$

$$= 18.51668$$

- Jarak ke bobot 2 (jenis objek kacang hijau)

$$= \sqrt{(107.96 - 72.28)^2 + (46.09 - 61.19)^2 + (26.06 - 23.08)^2 + (0.59}$$

$$= 38.85832$$

- Jarak ke bobot 3 (jenis objek kacang kedelai)

$$= \sqrt{(107.96 - 141.39)^2 + (46.09 - 106.32)^2 + (26.06 - 69.15)^2 + (0.59}$$

$$= 81.25251$$

Jarak terpendek adalah pada bobot ke-1 dengan target kelas adalah 1, sehingga bobot ke-1 yang baru adalah:

$$w_{11} = w_{11} + Lr * (x_{11} - w_{11}) = 91.28 + 0.9 * (107.96 - 91.28) = 106.292$$

$$w_{12} = w_{12} + Lr * (x_{12} - w_{12}) = 38.05 + 0.9 * (46.09 - 38.05) = 45.286$$

$$w_{13} = w_{13} + Lr * (x_{13} - w_{13}) = 26.00 + 0.9 * (26.06 - 26.00) = 26.054$$

$$w_{14} = w_{14} + Lr * (x_{14} - w_{14}) = 0.58 + 0.9 * (0.59 - 0.58) = 0.589$$

Sehingga diperoleh bobot 1 yang baru:

W1=[ ]

Iterasi 2:

Data ke-2 [113.67 49.59 28.19 0.60] mewakili kacang merah:

- Jarak ke bobot 1 (jenis objek kacang merah)

$$\sqrt{(113.67 - 106.292)^2 + (49.59 - 45.286)^2 + (28.19 - 26.054)^2 + (0.60 - 0.589)^2}$$

$$= 8.8046$$

- Jarak ke bobot 2 (jenis objek kacang hijau)

$$= \sqrt{(113.67 - 72.28)^2 + (49.59 - 61.19)^2 + (28.19 - 23.08)^2 + (0.60$$

$$= 44.2891$$

- Jarak ke bobot 3 (jenis objek kacang kedelai)

$$= \sqrt{(113.67 - 141.39)^2 + (49.59 - 106.32)^2 + (28.19 - 69.15)^2 + (0.60$$

$$= 75.26231$$

Jarak terpendek adalah pada bobot ke-1 dengan target kelas adalah 1, sehingga bobot ke-1 yang baru adalah:

$$w_{11} = w_{11} + Lr * (x_{21} - w_{11}) = 106.292 + 0.9 * (113.67 - 106.292) = 112.93$$

$$w_{12} = w_{12} + Lr * (x_{22} - w_{12}) = 45.286 + 0.9 * (49.59 - 45.286) = 49.159$$

$$w_{13} = w_{13} + Lr * (x_{23} - w_{13}) = 26.054 + 0.9 * (28.19 - 26.054) = 27.976$$

$$w_{14} = w_{14} + Lr * (x_{24} - w_{14}) = 0.589 + 0.9 * (0.60 - 0.589) = 0.598$$

Sehingga diperoleh bobot 1 yang baru:

W1=[ ]

Iterasi 3:

Data ke-3 [70.20 61.53 23.78 0.69] mewakili kacang hijau:

- Jarak ke bobot 1 (jenis objek kacang merah)

$$= \sqrt{(70.20 - 112.93)^2 + (61.53 - 49.159)^2 + (23.78 - 27.976)^2 + (0.69 -$$

$$= 44.6823$$

- Jarak ke bobot 2 (jenis objek kacang hijau)

$$= \sqrt{(70.20 - 72.28)^2 + (61.53 - 61.19)^2 + (23.78 - 23.08)^2 + (0.69$$

$$= 2.22101$$

- Jarak ke bobot 3 (jenis objek kacang kedelai)

$$= \sqrt{(70.20 - 141.39)^2 + (61.53 - 106.32)^2 + (23.78 - 69.15)^2 + (0.69$$

$$= 95.5646$$

Jarak terpendek adalah pada bobot ke-2 dengan target kelas adalah 2, sehingga bobot ke-2 yang baru adalah:

$$w_{61} = w_{61} + Lr * (x_{31} - w_{61}) = 72.28 + 0.9 * (70.20 - 72.28) = 70.408$$

$$w_{62} = w_{62} + Lr * (x_{32} - w_{62}) = 61.19 + 0.9 * (61.53 - 61.19) = 61.496$$

$$w_{63} = w_{63} + Lr * (x_{33} - w_{63}) = 23.08 + 0.9 * (23.78 - 23.08) = 23.71$$

$$w_{64} = w_{64} + Lr * (x_{34} - w_{64}) = 0.72 + 0.9 * (0.69 - 0.72) = 0.693$$

Sehingga diperoleh bobot 2 yang baru:

W_2=[ ]

Implementasi proses pelatihan LVQ dimulai dengan melakukan inisialisasi input, kelas target, dan parameter-parameter seperti maksimal epoh, *learning rate*, eror minimum yang diharapkan, serta nilai pengurangan *learning rate*. Inisialisasi yang dilakukan pada bahasa C# adalah sebagai berikut:

```
//inisialisi input training
double[,] x = new double[100, 5];
//inisialisi input testing
double[,] tes = new double[1, 4];
//inisialisasi bobot
double[,] w = new double[3, 4];
//jenis target
```

```
int[] C = { 1, 2, 3 };
int jenis;

//max epoh
int maxepoh;
int epoh;

//learning rate
int pilih_lr;
double lr;

//pengurangan learning rate
int pilih_dec_lr;
double dec_lr;

//eror minimum yg diharapkan
double emin = 0.01;

//euclidean distance
double ed1;
double ed2;
double ed3;
```

Untuk penentuan nilai awal bobot adalah 3 data pertama untuk setiap jenis objek. Pada pengambilan nilai bobot yang dilakukan pada bahasa C# adalah sebagai berikut:

```
int id = 0;
MySqlConnection connect = new MySqlConnection(connect_db);

string strSQL = "SELECT * FROM bobot";
MySqlCommand mysqlCmd = new MySqlCommand(strSQL, connect);
connect.Open();
MySqlDataReader reader;
reader = mysqlCmd.ExecuteReader();
while (reader.Read())
{
  w[id, 0] = (double)reader["red"];
  w[id, 1] = (double)reader["green"];
  w[id, 2] = (double)reader["blue"];
  w[id, 3] = (double)reader["roundness"];
  id = id + 1;
}
connect.Close();
```

Setelah itu dilakukan pengambilan data *training*. Pengambilan data *training* dari basis data adalah membaca setiap *record* kemudian menyimpannya dalam suatu array. Implementasinya pada bahasa C# dilakukan sebagai berikut:

```
int id = 0;
MySqlConnection connect = new MySqlConnection(connect_db);

string strSQL = "SELECT * FROM ciri";
MySqlCommand mysqlCmd = new MySqlCommand(strSQL, connect);
connect.Open();
```

```
MySqlDataReader reader;
reader = mysqlCmd.ExecuteReader();
while (reader.Read())
{
    x[id, 0] = (double)reader["r"];
    x[id, 1] = (double)reader["g"];
    x[id, 2] = (double)reader["b"];
    x[id, 3] = (double)reader["rd"];
    //load category
    x[id, 4] = (int)reader["j"];
    id = id + 1;
}
connect.Close();
```

Dengan telah dilakukan inisialisasi dan pengambilan data *training* dari *database*, maka pelatihan LVQ dapat dilakukan. Pelatihan berlangsung sesuai jumlah maksimal epoh dan *learning rate* seperti berikut ini:

```
while (epoh < maxepoh && lr > emin)
{
    epoh = epoh + 1;
    for (int i = 0; i <= 1; i++)
    {
        latih(i);
    }
    lr = lr - (lr * dec_lr);
}
```

Pada fungsi *latih()* proses yang terjadi berupa penghitungan jarak *euclidean*, kemudian melakukan perubahan nilai bobot sesuai target kelasnya. Berikut ini adalah proses penghitungan jarak *euclidean* pada bahasa C#:

```
for (int j = 0; j < 4; j++)
{
    ed1 = ed1 + ((x[a, j] - w[0, j]) * (x[a, j] - w[0, j]));
    ed2 = ed2 + ((x[a, j] - w[1, j]) * (x[a, j] - w[1, j]));
    ed3 = ed3 + ((x[a, j] - w[2, j]) * (x[a, j] - w[2, j]));
}

//akar
ed1 = Math.Sqrt(ed1);
ed2 = Math.Sqrt(ed2);
ed3 = Math.Sqrt(ed3);
```

Proses perubahan salah satu nilai bobot jika sesuai dengan kelasnya pada bahasa C# adalah sebagai berikut:

```
//ubah bobot w1
for (int i = 0; i <= 3; i++)
{
    w[0, i] = w[0, i] + lr * (x[a, i] - w[0, i]);
}
connect.Open();
string strSQL = "UPDATE bobot SET red='" + w[0,0] + "',green='" + w[0,1] +
"',blue='" + w[0,2] + "',roundness='" + w[0,3] + "' WHERE id_bobot='1'";
MySqlCommand mysqlCmd = new MySqlCommand(strSQL, connect);
```

```
mysqlCmd.ExecuteNonQuery();
connect.Close();
```

Proses perubahan salah satu nilai bobot jika tidak sesuai dengan kelasnya pada bahasa C# adalah sebagai berikut:

```
//ubah bobot w1
for (int i = 0; i <= 3; i++)
{
  w[0,i] = w[0,i] - lr * (x[a, i] - w[0,i]);
}
connect.Open();
string strSQL = "UPDATE bobot SET red='" + w[0,0] + "',green='" + w[0,1] +
"',blue='" + w[0,2] + "',roundness='" + w[0,3] + "' WHERE id_bobot='1'";
MySqlCommand mysqlCmd = new MySqlCommand(strSQL, connect);
mysqlCmd.ExecuteNonQuery();
connect.Close();
```

4.2.6 Segmentasi

Proses segmentasi ini hanya dijalankan pada fase kedua saja yaitu pada fase pengujian, segmentasi ini dilakukan dikarenakan gambar *input* yang diberikan adalah sebuah gambar yang merepresentasikan objek secara jamak, atau lebih dari satu. Tujuan segmentasi untuk memisahkan antara citra objek satu dengan yang lain agar memudahkan saat proses identifikasi, proses segmentasi ini meliputi proses pelabelan dan *cropping*.

4.2.6.1 Labeling

Proses labeling menggunakan metode *Connected Component Labeling* yang dilakukan dengan memanfaatkan beberapa fitur komponen *Filters* yang dibuat oleh Andrew Kirillov (*developer* Aforge *framework*) sebagai antar muka perangkat keras. *Filters* adalah salah satu pustaka dari *Aforge.Imaging* yang banyak digunakan untuk pengolahan citra digital. Dari pustaka *Filters* terdapat *class* bernama *ConnectedComponentsLabeling* yang diterapkan dalam aplikasi ini.

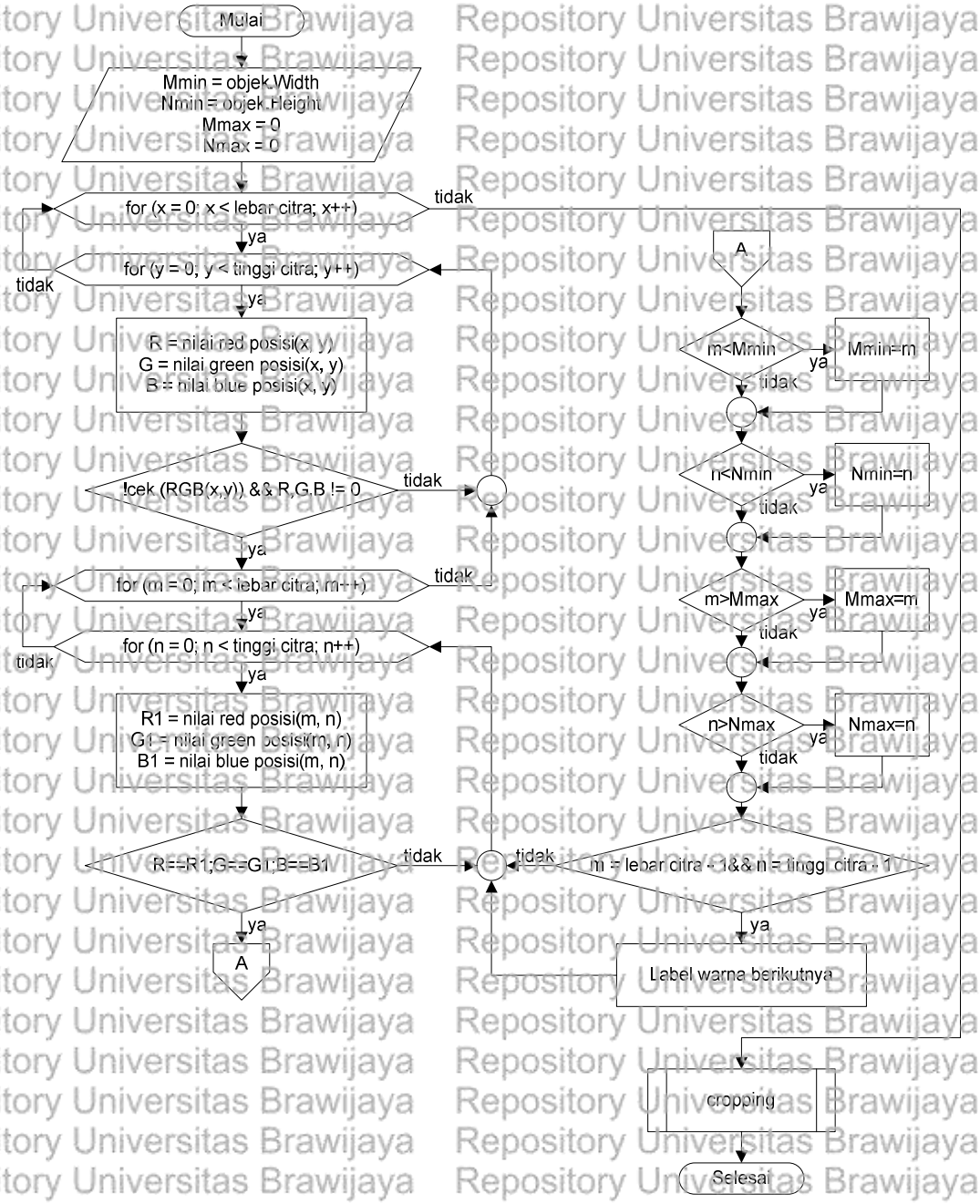
Proses pelabelan ini bisa difungsikan juga sebagai penghitung objek dalam citra, untuk penerapan dalam bahasa C# sebagai berikut:

```
ConnectedComponentsLabeling segmen = new ConnectedComponentsLabeling();
pictureBox22.Image = segmen.Apply(cocola);
int hitung = segmen.ObjectCount;
```

Hasil dari proses pelabelan tersebut adalah tiap-tiap objek memiliki warna yang berbeda sesuai dengan label yang didapat sesuai dengan proses labeling.

4.2.6.2 Cropping

Cropping atau pemotongan adalah proses yang dilakukan untuk memisahkan suatu citra menjadi citra baru sesuai dengan batas-batas piksel yang telah ditentukan. Dalam hal ini citra yang akan dipotong adalah citra objek dari gambar objek jamak menjadi citra objek individu, untuk batas-batas pemotongan menggunakan hasil pelabelan yang telah dilakukan pada tahap sebelumnya. Alur *cropping* dijelaskan pada Gambar 4.13.



Gambar 4.13 Flowchart *cropping*
(**Sumber:** perancangan)

Gambar 4.13 menjelaskan tentang proses *cropping* mulai dari seleksi warna label, penentuan koordinat hingga peletakan objek *cropping*.

Pada tahap sebelumnya objek telah diberi label dengan warna sebagai pembedanya, maka proses pengambilan objek dimulai dengan menentukan warna awal objek yang akan diambil. Ketika warna ditemukan, maka melakukan pengecekan apakah warna ini sudah tersimpan atau belum, jika belum maka proses iterasi akan berlanjut, berikut implementasi pada bahasa C# sebagai berikut:

```
for (int x = 0; x < objek.Width; x++)
{
  for (int y = 0; y < objek.Height; y++)
  {
    int R = (objek.GetPixel(x, y).R);
    int G = (objek.GetPixel(x, y).G);
    int B = (objek.GetPixel(x, y).B);

    warna = Color.FromArgb(R, G, B).ToArgb();

    if (!cek(objek.GetPixel(x, y)) && warna != Color.Black.ToArgb())
    {
      for (int m = 0; m < objek.Width; m++)
      {
        for (int n = 0; n < objek.Height; n++)
        {
          int R1 = (objek.GetPixel(m, n).R);
          int G1 = (objek.GetPixel(m, n).G);
          int B1 = (objek.GetPixel(m, n).B);
          warna1 = Color.FromArgb(R1, G1, B1).ToArgb();
```

Pada saat objek ditemukan iterasi tetap berjalan dan proses yang terjadi berikutnya yaitu menentukan koordinat yang akan digunakan untuk melakukan *cropping*, koordinat ini sebelumnya sudah didefinisikan dengan variabel sebagai berikut

```
          int Mmin = objek.Width;
          int Nmin = objek.Height;
          int Mmax = 0;
          int Nmax = 0;
```

Kemudian dilakukan proses pengecekan nilai (n,m) baru sebagai penentu koordinat *cropping* yang didapatkan dari kondisi sebagai berikut:

```
          if (R == R1 && G == G1 && B == B1)
          {
            if (m < Mmin) Mmin = m;
            if (n < Nmin) Nmin = n;
            if (m > Mmax) Mmax = m;
            if (n > Nmax) Nmax = n;
          }
```

Proses penetuan koordinat untuk objek pertama selesai jika iterasi sudah mencapai batas piksel maksimal, jika telah sampai pada batas piksel maksimal maka

warna yang sebelumnya merepresentasikan sebagai objek disimpan dan digunakan sebagai acuan untuk menemukan objek yang baru. Kemudian proses mencari objek dilakukan mulai dari iterasi awal lagi, dan proses ini berhenti ketika tidak lagi ditemukan warna baru. Implementasi penyimpanan warna sebagai berikut.

```
if (m == objek.Width - 1 && n == objek.Height - 1)
{
  warna_obj[counter_warna] = objek.GetPixel(x, y);
  counter_warna++;
}
```

Setelah proses iterasi selesai sepenuhnya selanjutnya proses *cropping* dilakukan menggunakan *class* bernama *Crop* dengan menggunakan parameter nilai-nilai Mmin, Nmin, Mmax, Nmax.

```
Crop detectionObyek = new Crop(new Rectangle(Mmin-5, Nmin-5, (Mmax - Mmin)+10,
(Nmax - Nmin)+10));
```

Terakhir yaitu peletakan objek yang telah melalui proses *cropping* dengan menerapkan picturebox yang telah disediakan.

```
if (pictureBox6.Image == null)
{ pictureBox6.Image = crop_Img; pictureBox24.Image = pictureBox6.Image; }
else if (pictureBox7.Image == null)
{ pictureBox7.Image = crop_Img; pictureBox25.Image = pictureBox7.Image; }
else if (pictureBox8.Image == null)
{ pictureBox8.Image = crop_Img; pictureBox26.Image = pictureBox8.Image; }
else if (pictureBox9.Image == null)
{ pictureBox9.Image = crop_Img; pictureBox27.Image = pictureBox9.Image; }
else if (pictureBox10.Image == null)
{ pictureBox10.Image = crop_Img; pictureBox28.Image = pictureBox10.Image; }
```

4.2.7 Identifikasi objek

Pada tahap perancangan identifikasi objek berdasarkan warna dan bentuk dilakukan dengan cara mengukur dan membandingkan jarak suatu vektor input objek yang belum diketahui jenisnya dengan vektor bobot setiap jenis objek yang telah didapatkan melalui proses pelatihan.

$$d_{ij} = \sqrt{(x_{iR} - w_{jR})^2 + (x_{iG} - w_{jG})^2 + (x_{iB} - w_{jB})^2 + (x_{i\,ground} - w_{j\,ground})^2}$$

dengan:

d_{ij} = euclidean distance input(X_i) dengan bobot(W_j)

x_{iR} = nilai vektor input red ke-i

x_{iG} = nilai vektor input green ke-i

x_{iB} = nilai vektor input blue ke-i

X_{iround} = nilai vektor input round ke-1

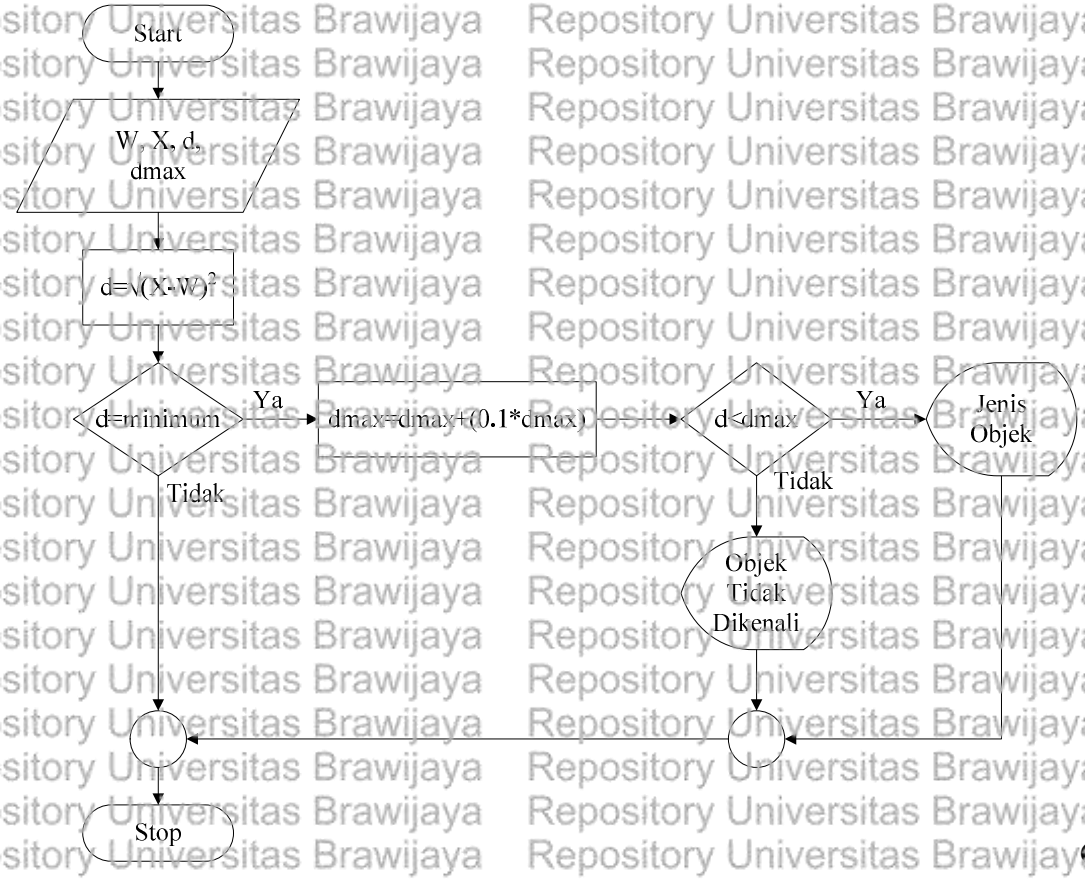
W_{iR} = nilai vektor bobot red ke-j

W_{iG} = nilai vektor bobot green ke-j

W_{iB} = nilai vektor bobot blue ke-j

W_{iround} = nilai vektor bobot round ke-j

Pengukuran jarak dilakukan dengan menggunakan metode euclidean distance. Jarak terkecil vektor objek input yang belum dikenali dengan vektor bobot jenis objek kemudian diperiksa apakah melebihi toleransi 10% jarak maksimum yang didapat dari pelatihan atau tidak. Jika jarak terkecil tersebut melewati nilai maksimum maka objek tidak akan dikenali, hal ini seperti yang dijelaskan pada gambar 4.14.



Gambar 4.14 Flowchart identifikasi Objek

(**Sumber:** Perancangan)

Implementasi proses identifikasi objek diawali dengan melakukan deklarasi variabel yang menyimpan perhitungan toleransi jarak. Deklarasi yang dilakukan pada bahasa C# adalah sebagai berikut:

```
double ed1max;
double ed2max;
double ed3max;
```

Selanjutnya adalah memanggil data *testing* yang berupa hasil ekstraksi ciri pada fase pengujian yang telah disimpan dalam array. Pemanggilan data *testing* yang terjadi pada bahasa C# adalah sebagai berikut:

```
tes[0, 0] = Rrata;
tes[0, 1] = Grata;
tes[0, 2] = Brata;
tes[0, 3] = roundness;
```

Setelah itu melakukan proses perhitungan jarak *euclidean* terhadap nilai bobot yang akan menentukan jenis daripada objek uji. Berikut ini adalah implementasi perhitungan jarak *euclidean* pada bahasa C#:

```
for (int j = 0; j <= 3; j++)
{
  ed1 = ed1 + ((tes[a, j] - w[0,j]) * (tes[a, j] - w[0,j]));
  ed2 = ed2 + ((tes[a, j] - w[1,j]) * (tes[a, j] - w[1,j]));
  ed3 = ed3 + ((tes[a, j] - w[2,j]) * (tes[a, j] - w[2,j]));
}
//akar
ed1 = Math.Sqrt(ed1);
ed2 = Math.Sqrt(ed2);
ed3 = Math.Sqrt(ed3);
```

Untuk menentukan jenis objek harus dilakukan suatu pemeriksaan jarak terkecil terhadap salah satu nilai bobot. Selain itu juga masih dilakukan pemeriksaan toleransi terhadap nilai jarak tersebut. Berikut ini adalah implementasi salah satu penentuan jenis objek pada bahasa C#:

```
if (ed1 < ed2 && ed1 < ed3)
{
  connect.Open();
  string strSQL = "SELECT MAX(ed1) AS ed1max FROM ed1";
  MySqlCommand mysqlCmd = new MySqlCommand(strSQL, connect);
  mysqlCmd.ExecuteNonQuery();
  MySqlDataReader reader;
  reader = mysqlCmd.ExecuteReader();
  while (reader.Read())
  {
    ed1max=(double)reader["ed1max"];
  }
  connect.Close();
  ed1max = ed1max + (ed1max * 0.1);
  if (ed1 <= ed1max)
  {
    km = km + 1;
    textBox8.Text = "Kacang Merah";
    label12.Text = "Kacang Merah";
  }
  else
  {
    td = td + 1;
```

```
        textBox8.Text = "Tidak Dikenali";
        label12.Text = "Tidak Dikenali";
    }
}
```

4.3 Implementasi Sistem

Setelah tahap perancangan tahap selanjutnya adalah tahap implementasi. Implementasi ini merupakan proses transformasi hasil perancangan perangkat lunak yang telah dibuat ke dalam kode *(coding)* sesuai dengan sintaks dari bahasa pemrograman yang digunakan.

4.3.1 Lingkungan Implementasi

Aplikasi dibuat dengan menggunakan Microsoft Visual Studio C# dengan Aforge .NET framework. Sistem diimplementasikan dengan menggunakan spesifikasi sebagai berikut:

1. Perangkat keras :

   A. Komputer

   Spesifikasi :

   - Processor : Intel Core™ 2 Duo *processor* T5870 2.00GHz
   - Memory : 1016 MB RAM
   - OS : Windows XP Service Pack 3
   - VGA : Mobile Intel 965 Express Chipset Family

2. Perangkat Lunak :

   - Sistem operasi : Microsoft Windows XP Service Pack
   - Bahasa pemrograman : Microsoft Visual Studio C# .NET 2010
   - Framework : Aforge .NET Framework
   - Basis data : MySQL
   - Tools : XAMPP 1.7.2

4.4 Implementasi Antarmuka

Aplikasi identifikasi dan penghitung berbagai jenis objek yang tercampur berdasarkan warna dan bentuk memiliki tampilan seperti pada gambar dibawah. Aplikasi ini memiliki 3 (tiga) tampilan yang menampilkan informasi berbeda sesuai fungsinya.

Gambar 4.15 Tampilan aplikasi pada tab Pelatihan

(**Sumber:** perancangan)

Pada gambar 4.15 adalah tampilan pada tab Pelatihan dimana terdapat 1 buah picturebox yang berfungsi untuk menampilkan citra objek input yang kita pilih guna dilakukan pelatihan. Untuk dapat melakukan penyimpanan data suatu objek, maka pengguna dapat menggunakan fasilitas tombol "Simpan Data" pada bagian tengah bawah. Bagian selanjutnya adalah pelatihan, dimana pengguna dapat melakukan proses pelatihan LVQ dengan mengatur terlebih dahulu nilai *maxepoh* dan *learning rate* dan *decreasing learning rate* kemudian menggunakan fasilitas tombol "Latih".

Gambar 4.16 Tampilan aplikasi pada tab Identifikasi

(**Sumber:** perancangan)

Pada gambar 4.16 adalah tampilan pada tab Identifikasi. Pada tab ini terdapat 2 buah picturebox, sebelah kiri berfungsi untuk menampilkan citra input yang akan diidentifikasi, sedangkan untuk yang sebelah kanan yaitu menampilkan hasil *cropping* (pemotongan) yang dilakukan pada gambar input.

Pada tahap identifikasi terdapat urutan proses yang harus dilakukan secara urut mulai dari pemisahan objek dengan background menggunakan fasilitas tombol "Pisah", kemudian *cropping* menggunakan fasilitas tombol "Cropping", dan terakhir mengidentifikasi objek yang telah di-*cropping* menggunakan fasilitas tombol "Identifikasi".

Gambar 4.17 Tampilan aplikasi pada tab Informasi

(**Sumber:** perancangan)

Setelah tombol "Identifikasi" pada tab Identifikasi ditekan maka secara otomatis tampilan akan berpindah menuju tab Informasi seperti yang ditunjukkan pada Gambar 4.17. Pada tab tersebut terlihat 1 buah picturebox yang menampilkan input citra identifikasi dan groupbox berlabel Informasi yang akan menampilkan citra objek yang telah melalui tahap *cropping* dan nama dari tiap-tiap jenis objek akan ditampilkan pada bagian samping tiap objek tersebut.