BAB 5 IMPLEMENTASI

5.1 Batasan Implementasi

Batasan implementasi dibutuhkan sebagai batasan dalam melakukan implementasi sehingga sesuai dengan perancangan dan untuk memperjelas ruag lingkupnya. Batasan dalam implementasi program pada penelitian ini adalah:

1. Teknik pengenalan emosi berdasarkan ekspresi mikro yang digunakan ialah dengan menerapkan ekstraksi fitur LBP dan metode klasifikasi K-NN.
2. *Pre-processing* citra ekspresi mikro seperti *Haar Cascade* untuk mengambil area wajah dibuat dengan memanfaatkan *library* dari *OpenCV*.
3. Citra ekspresi mikro yang digunakan sebagai data latih dan data uji merupakan citra dari basis data CASME II.
4. Citra ekspresi mikro yang digunakan sebagai data merupakan citra puncak ekspresi mikro.
5. Citra puncak ekspresi mikro merupakan citra berwarna.

5.2 Implementasi Algoritme

Algoritme yang diimplementasikan ke dalam program yaitu seperti yang telah dijelaskna pada perancangan, diantaranya ialah *pre-processing* citra, *processing* ekstraksi fitur LBP, klasifikasi dengan menggunakan metode K-NN.

5.2.1 Algoritme *Pre-processing* Citra Ekspresi Mikro

Pre-processing citra merupakan proses awal dalam pengolahan citra dalam implementasi program dengan masukan berupa citra *closed-up* berwarna. Pada algoritme *pre-processing* citra terdapat beberapa proses yang dilakukan yaitu ubah citra berwarna menjadi citra keabuan dan mengambil area wajah dengan menggunakan fungsi *Haar Cascade*. Citra keabuan didapat dari penjumlahan nilai *red, green, blue* pada gambar kemudian dibagi tiga. Citra keabuan digunakan untuk pencarian wajah menggunakan fungsi *Haar Cascade*. Setelah bagian wajah ditemukan dengan fungsi *Haar Cascade*, koordinat yang dinyatakan sebagai area wajah diatur dengan cara menambah atau mengurangi koordinat yang didapat sehingga citra yang didapatkan benar-benar hanya area wajah.

Algoritme 1: Tahap *pre-processing*

```
 1  # mengubah citra berwarna menjadi citra keabuan
 2  Masukan:
 3        img: citra berwarna
 4        b: panjang citra berwarna
 5        k: lebar citra berwarna
 6  Keluaran:
 7        abu: citra keabuan
 8
 9  abu ← array 2 dimensi bernilai 0 dengan panjang dan lebar (b, k)
10  FOR baris ← 0 TO b
11    FOR kolom ← 0 TO k
12      blue ← img[baris,kolom,0]
13      green ← img[baris,kolom,1]
14      red ← img[baris,kolom,2]
15      olah ← (red+green+blue)/3
```

```
Algoritme 1: Tahap pre-processing
16      abu[baris, kolom] ← olah
17    END-FOR
18  END-FOR
19
20  abu ← ubah tipe data menjadi uint8
21
22  #Mengambil area wajah menggunakan Haar Cascade
23  Masukan:
24        gray_img: citra keabuan
25  Keluaran:
26        wajah_abu: citra wajah keabuan
27
28  face_cascade ← mengambil Cascade Classifier berupa file xml
29  gray_img ← ubah tipe data menjadi unit8
30  faces ← melakukan pencarian wajah(gray_img, 1.3, 4)
31  wajah_abu ← salin citra gray_img
32  FOR (x, y, w, h) IN faces
33     wajah_abu ← resize(w, h)
34     wajah_abu ← ambil koordinat x,y sepanjang w,h pada
35                    gray_img
36  END-FOR
```

Penjelasan *pseudocode pre-processing* citra pada Algoritme 1:

Bagian mengubah citra berwarna menjadi citra keabuan.

1. Baris 9 inisialisasi variabel abu dengan array 2 dimensi yang bernilai 0 yang memiliki panjang dan lebar (*b, k*).
2. Baris 10-18 perulangan untuk melakukan proses perhitungan dari citra berwarna menjadi citra keabuan. Perulangan dilakukan dari awal piksel sampai piksel terakhir citra. Kemudian mengambil nilai *blue, green, red* dari citra berwarna, lalu dijumlahkan dan dibagi 3. Ganti nilai pada baris kolom saat ini pada variabel abu dengan nilai olah.
3. Baris 20 merubah tipe data abu menjadi uint8 agar bisa ditampilkan sebagai citra oleh sitem

Bagian mengambil area wajah menggunakan *Haar Cascade*.

1. Baris 28 mengambil *classifier* berupa file xml yang telah disediakan *OpenCV*.
2. Baris 29 mengubah tipe data masukan menjadi uint8 agar dapat diolah.
3. Baris 30 melakukan pencarian wajah dari citra masukan dengan parameter *scaleFactor* 1.3 dan *minNeghbors* 4.
4. Baris 31 inisialisasi variabel wajah_abu dengan nilai sama dengan citra masukan.
5. Baris 32 melakukan perulangan berdasarkan *x,y,w,h* yang didapat dari proses pencarian wajah.
6. Baris 33 *resize* array wajah_abu dengan ukuran $w \times h$.
7. Baris 34 ambil koordinat (*x,y)* yang telah ditetapkan sebagai area wajah dari citra masukan sebagai titik piksel awal dan *w,h* untuk panjang dan lebar dari citra masukan yang diambil.

5.2.2 Algoritme *Process* Ekstraksi Fitur LBP

Pada proses ekstraksi fitur LBP terdapat beberapa proses di dalamnya yaitu mengambil delapan nilai tetangga dari piksel yang ingin diolah, merubah nilai pada piksel tetangga menjadi nilai biner, menyusun menjadi deret bilangan biner, ubah deret bilangan biner menjadi angka desimal.

Algoritme 2: Ekstraksi Fitur LBP

```
#ekstaksi fitur LBP
Masukan:
       gray_img: citra wajah keabuan
       b: panjang baris piksel gray_img
       k: lebar kolom piksel gray_img
       r: jari-jari
       p: jumlah ketetanggan
Keluaran:
       lbinaryp: citra LBP

lbinaryp ← array 2 dimensi bernilai 0 dengan panjang b dan lebar k
IF r=2 AND p=8
  FOR baris ← r TO (b-r)
    FOR kolom ← r TO (k-r)
      array ← inisialisasi variabel bertipe array
      hitung ← inisialisasi variabel bernilai 0
      array ← memasukan nilai piksel gray_img[baris][kolom] pada
array

      u ← -2
      FOR x ← 0 TO 3
        array ← memasukan nilai piksel gray_img[baris-r][kolom+u]
         u ← u+2
      END-FOR

      v ← 0
      FOR y ← 0 TO 2
        array ← memasukan nilai piksel gray_img[baris+v][kolom+r]
         v ← v+2
      END-FOR

      w ← 0
      FOR x ← 0 TO 2
        array ← memasukan nilai piksel gray_img[baris+r][kolom+w]
         w ← w+2
      END-FOR
      array ← memasukan nilai piksel gray_img[baris+r][kolom+w]
      biner ← inisialisasi varibel array untuk meyimpan deret bil.
Biner

      FOR i ← 0 TO (len(array)-1)
        IF array[i+1] < array[0]
          biner ← masukan nilai 0 pada array
        ELSE
          biner ← masukan nilai 1 pada array
        END-IF
      END-FOR

      z ← 7
      FOR l ← 0 TO 8
        hitung ← hitung + 2z * biner[l]
        z ← z-1
      END-FOR

      lbinary[baris][kolom] ← hitung

```

```
Algoritme 2: Ekstraksi Fitur LBP
57  ELSE
58    FOR baris ← r TO (b-r)
59      FOR kolom ← r TO (k-r)
60        array ← inisialisasi variabel bertipe array
61        hitung ← inisialisasi variabel bernilai 0
62        array ← memasukan nilai piksel gray_img[baris][kolom] pada
63  array
64
65        FOR x ← -r TO (r+1)
66          array ← memasukan nilai piksel gray_img[baris-r][kolom+x]
67        END-FOR
68
69        FOR y ← -(r-1) TO r
70          array ← memasukan nilai piksel gray_img[baris+y][kolom+r]
71           v ← v+2
72        END-FOR
73
74        turun ← r
75        FOR x ← -r TO (r+1)
76          array ← memasukan nilai piksel gray_img[baris+r][kolom+w]
77           turun ← turun-1
78        END-FOR
79
80        FOR y ← -(r-1) TO r
81          array ← memasukan nilai piksel gray_img[baris+r][kolom+w]
82        END-FOR
83
84        biner ← inisialisasi varibel array untuk meyimpan deret bil.
85  Biner
86
87         FOR i ← 0 TO (len(array)-1)
88          IF array[i+1] < array[0]
89            biner ← masukan nilai 0 pada array
90          ELSE
91            biner ← masukan nilai 1 pada array
92          END-IF
93
94        z ← 7
95        FOR l ← 0 TO 8
96          hitung ← hitung + 2z * biner[l]
97          z ← z-1
98        END-FOR
99
100       lbinary[baris][kolom] ← hitung
101 END-IF
```

Penjelasan *pseudocode processing* pada Algoritme 2:

1. Baris 13-14 dan 58-59 perulangan dimulai dari indeks ke *r* dan berakhir pada *b-r*. pengambilan indeks yang diolah tidak dimulai dari paling awal karena akan menyebabkan kondisi *error* keluar dari batas, kondisi awal mengikuti dari nilai *r* agar mendapatkan piksel-piksel tetangga. Karena tidak dimulai dari awal maka juga perulangan tidak mengecek sampai piksel terakhir, tetapi berakhir pada indek terakhir dikurangi nilai *r*.
2. Baris 20-36 dan 65-82 merupakan proses pencarian ketetanggan.
3. Baris 41-47 dan 87-92 merupakan proses membandingkan piksel saat ini dengan piksel tetangganya.

4. Baris 49-53 dan 94-98 merupakan proses mengubah bilangan biner menjadi bilangan decimal.

5.2.3 Algoritme *Process* Klasifikasi K-NN

Pada proses klasifikasi masukan yang digunakan berupa nilai histogram LBP yang telah didapatkan dari proses ekstraksi fitur LBP dengan keluaran yang dihasilkan adalah kelas atau hasil pengenalan emosi.

```
Algoritme 3: klasifikasi K-NN
1   #menghitung jarak Euclidean
2   Masukan:
3         training: fitur citra data latih
4         test: fitur citra data uji
5         length: panjang fitur citra data uji
6   keluaran:
7         jarak: jarak Euclidean
8
9   jarak ← inisialisasi variabel jarak dengan nilai 0
10  FOR x ← 0 TO length
11    jarak ← jarak + sqrt((training[x] – test[x])²)
12  END-FOR
13
14  #menghitung jarak Manhattan
15  Masukan:
16        training: fitur citra data latih
17        test: fitur citra data uji
18        length: panjang fitur citra data uji
19  keluaran:
20        jarak: jarak Manhattan
21
22  jarak ← inisialisasi variabel jarak dengan nilai 0
23  FOR x ← 0 TO length
24    jarak ← jarak + absolut(training[x] – test[x])
25  END-FOR
26
27  #menghitung jarak Chebyshev
28  Masukan:
29        training: fitur citra data latih
30        test: fitur citra data uji
31        length: panjang fitur citra data uji
32  keluaran:
33        jarak: jarak Chebyshev
34
35  jarak ← inisialisasi variabel jarak dengan nilai 0
36  FOR x ← 0 TO length
37    jarak ← absolut(training[x] – test[x])
38  END-FOR
39  jarak ← max(jarak)
40
41  #klasifikasi K-NN
42  Masukan:
43        training: fitur citra data latih
44        test: fitur citra data uji
45        k: banyak tetangga terdekat
46  keluaran:
47        hasil: kelas terpilih
48
49  distance ← inisialisasi variabel array untuk menyimpan nilai jarak
50  length ← panjang fitur dari citra data uji
51  FOR x ← 0 TO panjang fitur citra data latih
52    dist ← perhitungan jarak
```

```
53    distance ← memasukan fitur-fitur data latih dan hasil
54  perhitungan jarak
55  END-FOR
56  distance ← mengurutkan berdasar nilai jarak dari yang terkecil
57  neighbors ← inisialisasi variabel array untuk menyimpan tetangga-
58  tetangga terdekat
59
60  FOR x ← 0 TO k
61    neighbors ← masukan tetangga terdekat dari variabel distance
62  END-FOR
63
64  classvotes ← untuk menyimpan kelas yang sering muncul
65  FOR x ← 0 TO k
66    response ← menyimpan nama kelas ke-x pada neighbors
67    IF nama kelas yang ada di response ADA pada classvotes
68      classvotes ← jumlah kelas diditambah 1
69    ELSE
70      classvotes ← response
71    END-IF
72  END-FOR
73
74  sortedVotes ← mengurutkan kelas yang sering muncul
75  hasil ← max(sortedVotes)
```

Penjelasan *pseudocode* klasifikasi K-NN pada Algoritme 3:

1. Baris 49-55 merupakan proses menghitung jarak antara data latih dan data uji. Perhitungan jarak dapat menggunakan algoritme pada baris 1-12 atau 14-25 atau 27-39
2. Baris 56-62 merupakan proses pengurutan hasil perhitungan jarak dan pencarian tetangga terdekat sesuai penentuan nilai variabel *k* sebagai banyaknya tetangga terdekat.
3. Baris 64-72 merupakan proses untuk menghitung kelas yang sering muncul.
4. Baris 74 merupakan poses pengurutan berdasarkan kelas yang paling sering muncul.
5. Baris 75 mengambil 1 kelas yang paling sering muncul sebagai hasil klasifikasi.