BAB III
METODOLOGI PENELITIAN

3.1 Analisis Kebutuhan

Dalam pembuatan aplikasi ini dibutuhkan aspek pendukung, yaitu perangkat keras (*hardware*) dan perangkat lunak (*software*) seperti berikut ini.

A. Perangkat Keras (*Hardware*)

Perangkat keras merupakan seluruh komponen atau unsur peralatan yang digunakan sebagai penunjang pembangunan suatu sistem. Adapun spesifikasi minimum perangkat keras yang diperlukan meliputi :

1. Kompatibel dengan Pentium(R) Core 2 Duo CPU
2. Space harddisk yang diperlukan minimal 500 MB (diluar OpenCV dan CodeBlocks).
3. RAM minimal 1GB.

B. Perangkat Lunak (*Software*)

Analisis perangkat lunak cenderung lebih menekankan kepada aspek pemanfaatan sumber daya (*software*). Spesifikasi perangkat lunak yang diperlukan untuk membangun aplikasi diantaranya:

1. *Windows XP (32 bit)/ Windows Vista (32 atau 64 bit)/ Windows 7 (32 atau 64 bit), Mac OS X 10.5.8* atau yang lebih tinggi, *Linux*.

2. *OpenCV*

   OpenCV digunakan sebagai *library* dalam pengolahan citra digital.

3. *MinGW*

   MinGW adalah *compiler* yang digunakan untuk membangun *library Opencv* dan sebagai *compiler* aplikasi.

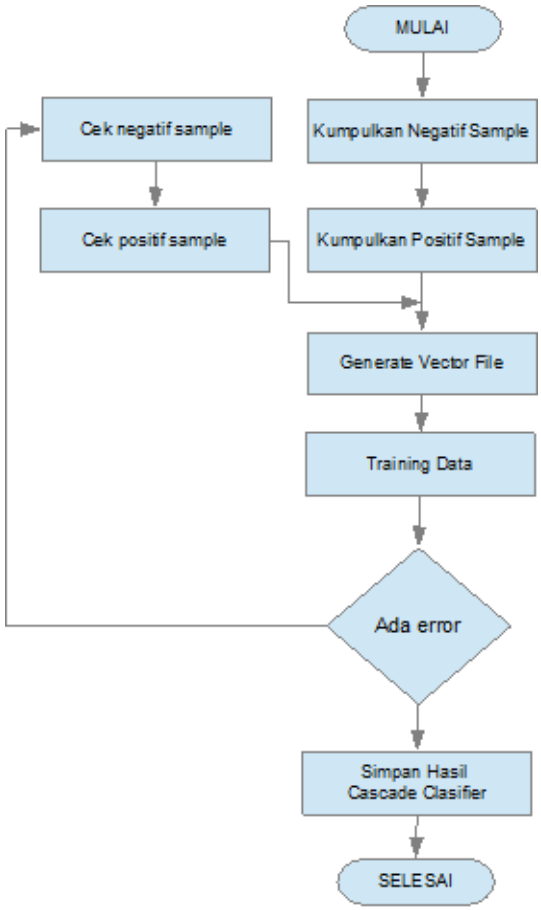
4. *Code Blocks*

   CodeBlocks adalah *IDE* yang sering digunakan sebagai program untuk membangun aplikasi atau program pada *windows*.

3.2 *Flowchart* Alur Sistem Presensi

Gambar 3.1 Flowchart Alur Sistem Presensi

3.3 Tahapan *Training Cascade clasifier*



Gambar 3.2 Training Cascade Clasifier

3.3.1 Kumpulkan negatif *sample*

Negatif sample adalah citra yang tidak mengandung objek yang akan dideteksi yaitu dalam kasus ini adalah objek wajah. Pada *opencv* berikut ini hal yang harus dilakukan untuk mengumpulkan negatif *sample*:

- Buat file **.txt*, misal *negatif.txt*
- Buat folder img, semua gambar negatif *sample* letakan di *folder* ini
- Isi bg.txt dengan *folder* dan nama file gambar, misal *img/1.jpg*

3.3.2 Kumpulkan positif *sample*

Positif sample adalah citra yang mengandung objek yang akan dideteksi. *Positif sample* dapat dibuat dengan cara menandai secara manual bagian objek yang akan dideteksi. Contoh perintah dalam *opencv* :

```
opencv_createsamples -info annotations.lst -img wajah.png -
bg negatif.txt -num 2340 -bgcolor 0 -bgthresh 0 -maxxangle
1.1 -maxyangle 1.1 -maxzangle 0.5 -w 50 -h 50
```

3.3.3 *Generate vector file*

Pada proses ini dibuat berupa *vector file* yang berisi kombinasi *negatif* dan *postitif* sample. Contoh perintah dalam *opencv* :

```
opencv_createsamples -info annotations.lst -vec sample.vec -
num 2340 -bgcolor 0 -bgthresh 0 -maxxangle 1.1 -maxyangle
1.1 -maxzangle 0.5 -w 50 -h 50
```

3.3.4 *Training Data*

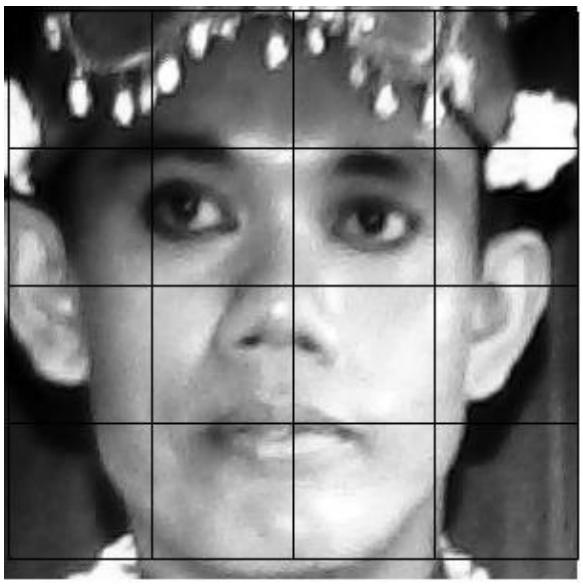
Proses ini memakan waktur relatif cukup lama, tergantung dari seberapa banyak data yang di proses dan banyaknya *stage* dalam pemrosesan. *Output* dari *training* ini berupa *cascadeclasifier.xml* yang dapat digunakan untuk mendeteksi objek wajah yang diinginkan. Dalam penelitian ini menggunakan *clasifier* yang sudah di sediakan oleh *OpenCV*. Contoh perintah *training data* dalam opencv :

```
opencv_traincascade -data clasifier -vec sample.vec -bg
negatif.txt -numPos 2000 -numNeg 1800 -numStages 12 -
precalcValBufSize 500 -precalcIdxBufSize 500 -featureType
LBP -w 50 -h 50 -minHitRate 0.999 -maxFalseAlarmRate 0.3
```

3.4 Tahapan Pengenalan dengan *Local Binary Pattern*

A. Membagi citra menjadi beberapa region/wilayah

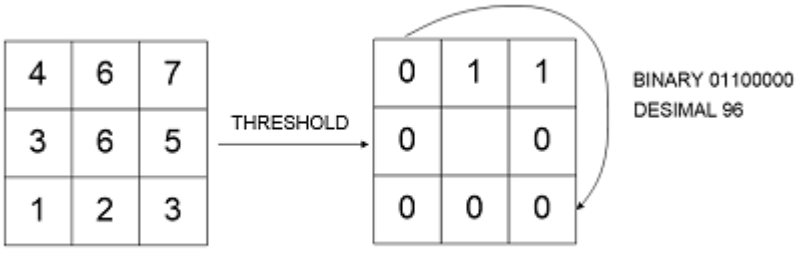
Proses ini membagi citra menjadi beberapa *block* atau wilayah. Misalnya suatu citra wajah di bagi menjadi *8 block* atau *16 block*.



Gambar 3.3 Citra yang dibagi menjadi 16 block

B. Masing-masing *region* dibuat kedalam bentuk *LBP*

Tahapan ini dilakukan dengan memakai fitur ketetanggaan *rectangle* 3x3.



Gambar 3.4 Fitur ketetanggaan 3x3

Pada langkah ini akan menggunakan nilai tengah sebagai pembanding dengan nilai tetangganya. Jika nilai tetangga < nilai tengah maka diubah menjadi 0 dan

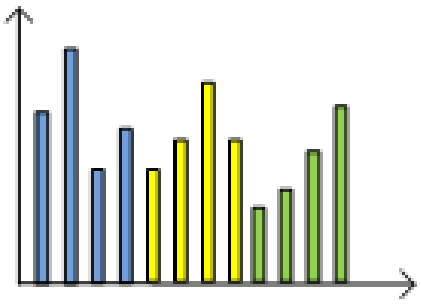
jika nilai tetangga >= nilai tengah maka diubah menjadi 1. Setelah semua tetangga berubah menjadi 0 atau 1 kemudian diubah kedalam *biner*. Data diambil dari kiri atas searah jarum jam. Nilai *binary*-nya diubah ke bentuk desimal dan menggantikan nilai tengah. Lakukan langkah ini terus menerus sampai semua *pixel* terlewati.

C. Buat *histogram* dari masing-masing *region*

Pada tahapan ini tiap *region* diubah kedalam bentuk histogramnya.

D. Gabungkan bentuk *histogram* dari tiap-tiap *region*

Setelah *histogram* terbentuk, gabungkan semua *histogram* yang ada menjadi satu.



Gambar 3.5 Histogram yang digabungkan menjadi satu

E. Tandai *histogram* tersebut sesuai dengan kelasnya.

F. Ulangi langkah sampai semua citra sudah diubah kedalam bentuk *histogram*

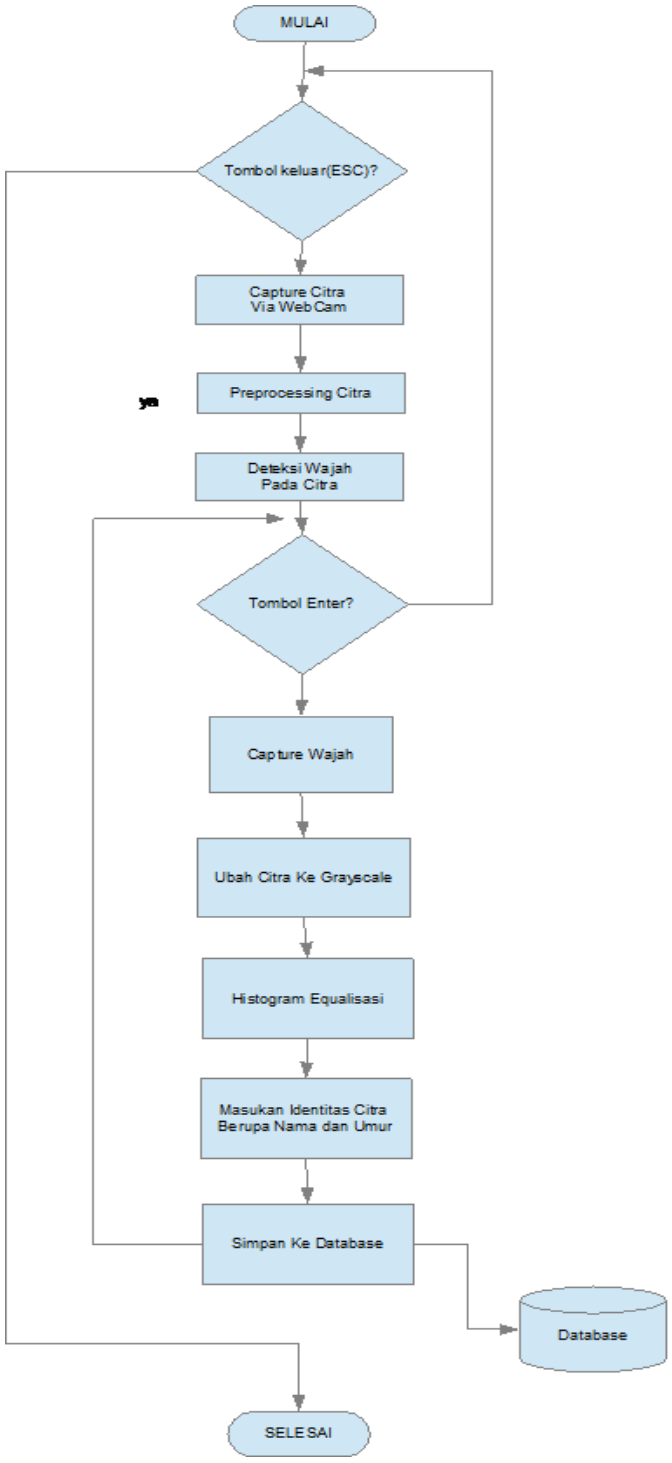
G. Untuk mencari kemiripan dalam model 2 dimensi bisa menggunakan *eucledian distance*

Misal *Histogram 1 Class A* = {3,4,5,2,3};

Misal *Histogram 1 Class B* = {1,2,5,2,3};

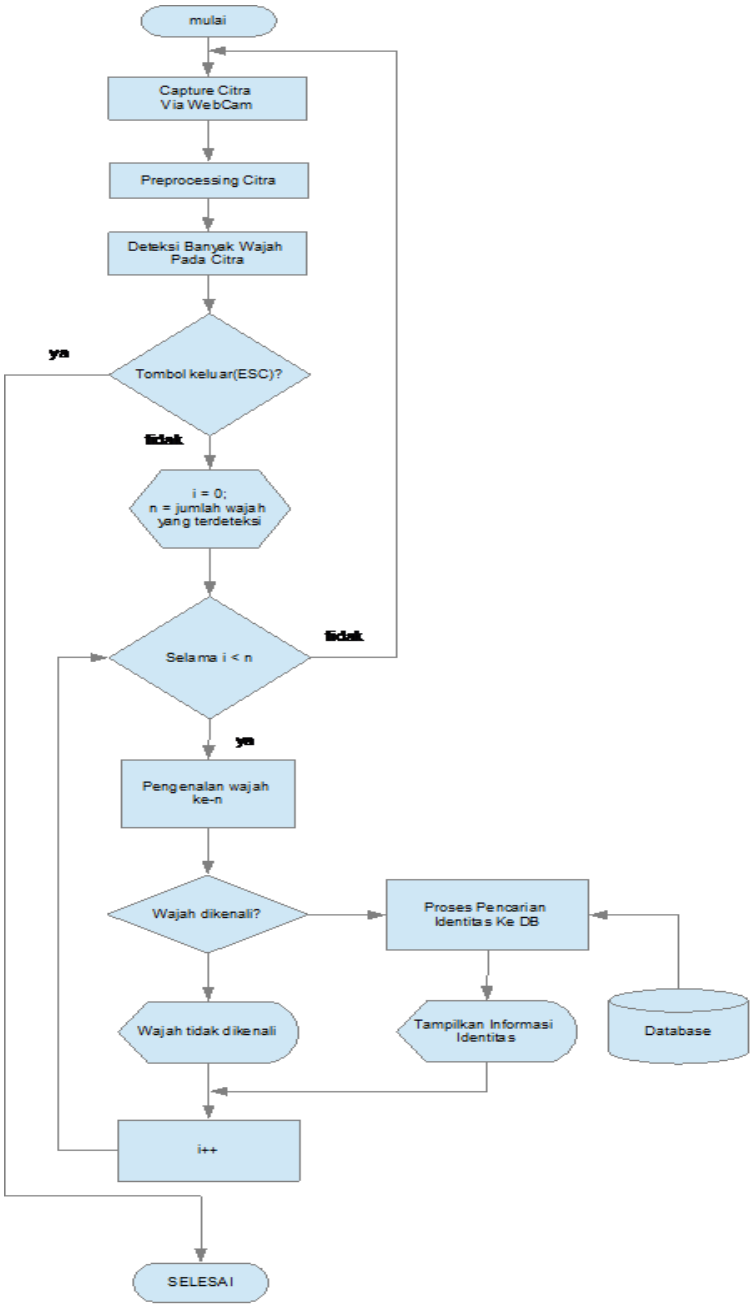
Sim (A,B) = 2.85

3.5 Tahapan Pengambilan Citra Wajah



Gambar 3.6 Tahapan pengambilan citra wajah

3.6 Tahapan Pengenalan *realtime*



Gambar 3.7 Tahapan pengenalan realtime