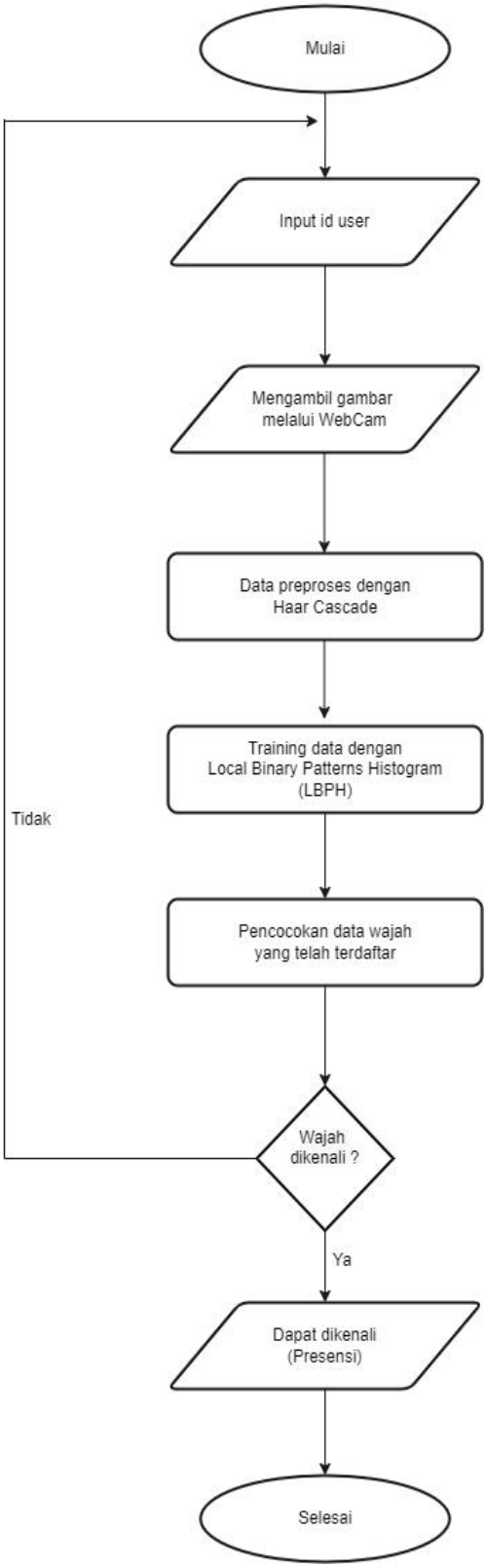
BAB 3
ANALISA DAN PERANCANGAN

Pada Bagian ini Bab ini akan membahas tentang analisa dan perancangan dalam sistem pengenalan wajah untuk melakukan kehadiran otomatis dalam aplikasi "*Smart Attendance Recognition*". dalam Bab ini terdapat beberapa tahapan yang akan dianalisa, lalu tahap-tahap selanjutnya adalah sebagai berikut. Tahapan yang pertama adalah analisa data yang akan digunakan. Lalu tahapan kedua adalah analisa proses dalam mengolah citra yang akan digunakan. Selanjutnya yang terakhir adalah tahapan ketiga adalah analisa implementasi metode *Haar Cascade Classifier* dan *Local Binary Pattern Histogram* (LBPH) dalam melakukan pengenalan wajah untuk kehadiran secara *real time*.

3.1 Gambaran Umum Alur Kerja Sistem

Metode yang akan digunakan dalam melakukan pengenalan wajah untuk kehadiran otomatis secara real time adalah dengan menggunakan metode *Haar Cascade Classifier* dan *Local Binary Pattern Histogram* (LBPH) dengan melewati bebeberapa tahapan. Tahapan-tahapan tersebut akan dimulai dengan membuat data citra. Tahapan selanjutnya adalah tahapan *pre-processing* yang terdiri dari proses *convert to grayscale* dan *resize,* proses ini akan mengubah citra agar proses *training* dapat lebih mudah dilakukan. Setelah tahap *preprocessing* sudah dilakukan, maka akan masuk ke dalam tahapan melakukan pengenalan wajah untuk kehadiran dengan mengimplementasi Metode *Haar Cascade Classifier* dan *Local Binary Pattern Histogram* (LBPH) . Setelah tahapan-tahapan tersebut dilakukan maka akan mendapatkan hasil pengenalan wajah user yang sudah terdaftar sebelumnya di *dataset*. Gambaran umum alur kerja dari sistem dapat dilihat melalui *flowchart* pada Gambar 3.1. berikut.



Gambar 3.1 Flowchart alur kerja sistem

Berikut uraian setiap proses yang terdapat pada *flowcharat,* yaitu :

1. Input Id User

   Merupakan sebuah *input* atau masukan pada sistem yang dilakukan secara *real time.* Untuk melakukan pengimputan ID yang diperlukan untuk inisialisasi setiap wajah yang akan direkam.

2. Mengcapture Video dari *Webcam*

   Merupakan sebuah *input* atau masukan pada sistem yang dilakukan secara *real time.* Untuk melakukan pendaftaran wajah diperlukan kamera atau *webcam* yang digunakan untuk merekam wajah.

3. Data *Preproses* dengan *Haar cascade*

   Langkah ini dilakukan untuk melakukan *transformasi* pada gambar yang telah berhasil direkam melalui kamera atau *webcam* menjadi bentuk *grayscale* dan sudah dilakukan *resize* sesuai dengan wajah yang terdeteksi lalu akan ditunjukkan berupa garis bujur sangkar ROI (*Region of Interest*) pada wajah yang berhasil dideteksi.

4. Training Data dengan *Local Binary Pattern Histogram* (LBPH)

   Langkah dalam mengenali wajah manusia adalah dengan mendeteksi wajah terlebih dahulu. Pada training wajah akan dilakukan proses pencocokan berdasarkan data yang sudah dilakukan *transformasi* sebelumnya dengan *haar cascade.*

5. Pencocokan Data Wajah

   Pencocokan wajah dilakukan untuk mengenali wajah manusia berdasarkan dataset wajah yang sudah ada. Wajah akan dicocokan menggunakan metode *Haar Cascade Classifier.* Pada tahap ini terdapat perhitungan-perhitungan untuk pengenalan wajah.

6. Wajah dikenali

   Wajah dikenali adalah sebuah kondisi apakah wajah manusia dikenali atau tidak. Berdasarkan *dataset* wajah yang sudah ada. Jika wajah dapat dikenali maka akan lanjut ke proses berikutnya, jika wajah tidak dikenali maka akan kembali ke proses sebelumnya yaitu Input Id *User*.

7. Dapat dikenali (Presensi)

Setelah melakukan pengujian dengan data wajah yang ada dalam *dataset* dan berhasil , sistem akan memberikan informasi nama wajah yang dikenali secara dan juga secara otomatis akan masuk kedalam file .csv yang berisi informasi berupa nama dan tanggal waktu melakukan presensi.

3.2 Data yang digunakan

Data yang akan digunakan dalam penelitian ini terdiri dari sampel gambar yang diambil dari hasil *capture* sebuah kamera *webcam* sebanyak 100 *capture* gambar per-wajah yang akan didaftarkan, dengan beberapa batasan parameter yaitu : variasi posisi citra wajah dan jarak wajah terhadap kamera *webcam*.

Untuk variasi posisi wajah dilakukan beberapa posisi sebagai berikut:

- Menghadap tegak lurus ke depan
- Rotasi 10° ke kanan
- Rotasi 10° ke kiri
- Rotasi 10° ke ke atas
- Rotasi 10° ke bawah
- Mengangkat dagu 10° atas

Wajah yang di *capture webcam* tidak terhalangi sebagaian oleh objek lain, tidak banyak terpotong dan tidak bergerak. Untuk aspek jarak wajah terhadap kamera webcam akan dicari jarak ideal yaitu 40 cm sampai dengan 100 cm.

3.3 Pembuatan *Dataset*

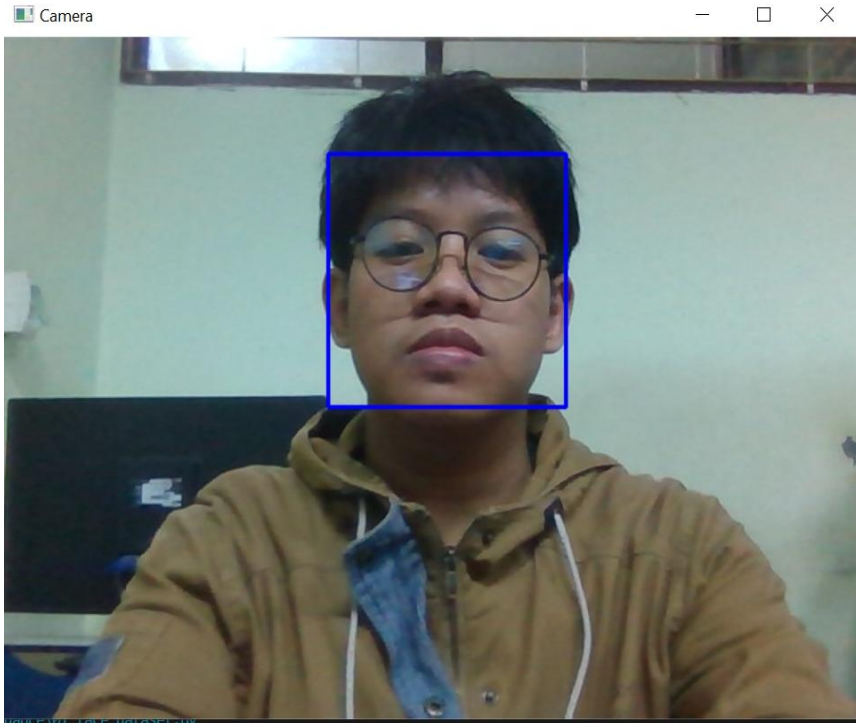
Sebelum sistem dapat mengenali wajah, wajah yang akan dikenali terlebih dahulu disimpan pada database agar sistem dapat mengetahui nilai *LBPH (Local Binary Patern Histogram)* atau nilai *histogram* dari sebuah *image*. Untuk dapat mengenali wajah dengan baik sistem memerlukan 100 input gambar dari kamera. Dari setiap subjek diambil datanya dengan berbagai pose dan sudut sebanyak 100 kali. Data tersebut akan disimpan sebagai *dataset* gambar. Pembuatan *dataset* diperlukan untuk mendapatkan data wajah yang akan dikenali. Sampel-sampel tersebut kemudian dimasukkan dalam sebuah folder dengan tiap orangnya memiliki nomor unik tersendiri.

Nomor tersebut digunakan untuk mengidentifikasi sampel wajah dari orang tersebut. Sampel gambar kemudian diubah menjadi gambar yang bersifat *grayscale* sebelum disimpan pada *dataset* .Setelah itu, dilakukan proses training. Pada proses ini gambar pada *dataset* akan diekstraksi nilai histogramnya. Nilai tersebut disimpan dalam bentuk data array dan disimpan beserta nomor identitas masing-masing subjek.

3.4 *Preprocessing*

Dalam proses preprocessing ini akan dilakukan tahap pengolahan citra agar data yang akan diolah mempunyai citra yang lebih baik untuk diproses ke tahapan selanjutnya. Pada tahap proses *preprocessing* ini terdiri dari 2 (dua) proses, yaitu proses *capture* gambar wajah *user* dan dikonversi dari citra RGB menjadi citra *grayscale* lalu dilakukan resize.

Hal ini berguna untuk membuat citra dapat diolah dengan menggunakan metode *Haar Cascade*, yang dimana metode ini akan digunakan untuk mendeteksi wajah. Setelah dapat terdeteksi wajah maka sistem akan menandai wajah dengan garis bujur sangkar ROI (*Region of Interest*) berwarna biru dan objek yang digunakan adalah muka. Hal ini dilakukan untuk membedakan dengan objek bukan wajah. Contoh pada tahapan ini akan digambarkan dalam Gambar berikut 3.2



Gambar 3.2 Pengambilan gambar

Lalu akan dijelaskan untuk tahapan-tahapan melakukan *preprocessing* seperti contoh pada gambar 3.3 agar dataset yang akan digunakan dapat lebih mudah dan lebih cepat diproses sebagai berikut.

3.4.1 Face Detection

Perancangan face detection atau biasa disebut dengan pendeteksian wajah pada tugas akhir ini menggunakan metode *Haar Cascade Classifier*. Metode *Haar Cascade Classifier* digunakan karena metode ini sangat ringan dan memiliki kecepatan pendeteksian yang cepat. Dengan menggunakan *Library* yang disediakan oleh Open CV yaitu *haarcascadefrontal_face_default.xml.* Library tersebut sudah dapat semua proses pendeteksian wajah dan berikut ini merupakan tahapan cara kerja *library* Open CV haarcascadefrontal_face_default.xml yaitu :

Langkah pertama citra harus dikonversikan dari RGB menjadi *Grayscale.* Secara teori cara mengkonversikannya adalah dengan cara menggunakan rumus sebagai berikut.

$$Grayscale = 0.2989\ R + 0.5870\ G + 0.1140\ B \qquad \textbf{(3.1)}$$

Pada persamaan 3.1 dapat melihat bahwa setiap nilai R nilai G dan nilai B pada setiap piksel akan dikalikan dengan nilai yang sesuai rumus agar sebuah citra dapat dikonversi menjadi *grayscale*. Maka contoh terdapat pada gambar 3.3 merupakan citra RGB dengan dimensi 200 x 200.

Gambar 3.3 Citra RGB

Kemudian akan menggunakan semua piksel secara acak untuk mengetahui nilai RGB dari salah satu piksel. Lalu pada piksel x = 100 dan y = 100 dapat diketahui RGB = 57 34 16. Dari data tersebut maka dapat dihitung *grayscale* dengan cara sebagai berikut ini :

Diketahui = *Red* (merah) = 57
Green(hijau) = 34
Blue(biru) = 16

Grayscale = 0.2989 R + 0.5870 G + 0.1140 B
Grayscale = *(* 0.2989 x 57) +(0.5870 x 34) + (0.1140 x 16)
Grayscale = 17.0373 + 19.968 + 1.824
Grayscale = 38.8193

Citra *grayscale* merupakan citra dengan ukuran 8 bit. Sehingga didapatkan $(2^8 – 1)$ yang mengubah warna mulai dari 0 hingga 255 , yaitu dimana 0 berarti hitam dan 255 berarti putih. Warna abu memiliki *range* dari 1 hingga 254 dimulai dari abu paling gelap sehingga abu terang dan akhirnya mendekati putih. Maka hasil *grayscale* = 38.8193 merupakan warna abu-abu yang hampir mendekati hitam. Lalu menggunakan rumus tersebut pada semua poksel dan hasil dari konversi citra RGB pada gambar 3.4 akan menjadi seperti pada gambar 3.4 sebagai berikut :

Gambar 3.4 Citra grayscale

Dari gambar 3.4 dapat diketahui bahwa setelah melakukan konversi menjadi citra *grayscale* sebagian besar daerah pada citra memiliki nilai yang akan lebih hitam. Proses selanjutnya adalah melakukan *Haar feature* yaitu adalah metode yang membutuhkan training terlebih dahulu untuk mendapatkan suatu keputusan apakah di *frame* tersebut dapat mendeteksi objek atau tidak. Selisih dari nilai fitur yang akan diimplementasikan akan dijadikan *threshold* klasifikasi terdeteksi objek atau tidak. Lalu dalam tugas akhir ini objek yang dideteksi adalah wajah. Berikut ini adalah contoh gambar 3.5 yang akan diletakan fitur *haar* dalam sebuah citra.

Gambar 3.5 Haar Feature

Selanjutnya pada gambar 3.3 terdapat sebuah *feature haar* pada gambar hasil konversi *grayscale. Feature* yang digunakan berupa *line feature* yang merupakan salah satu dari banyak *haar feature.* Untuk menentukan nilai feature tersebut dapat digunakan persamaan yaitu sebagai berikut ini :

NILAI FITUR = |(total piksel hitam) – (total piksel putih) | (3.2)

Pada persamaan 3.2 dibutuhkan nilai piksel, ada berbagai cara untuk mengetahui nilai piksel. Tetapi dalam kasus ini dibutuhkan teknik yang dapat dilakukan dengan sangat cepat sehingga dapat diimplementasikan pada ratusan fitur *haar* dengan skala yang berbeda-beda. Salah satu teknik yang efisien untuk melakukan itu yaitu *integral image.* Berikut merupakan rumus integral image :

$$S(x,y) = i(x,y) + S\big((x-1),y\big) + S\big(x,(y-1)\big) - S((x-1),(y-1)) \quad (3.3)$$

Pada persamaan 3.3 i(*x,y*) adalah nilai asli matrik citra. Pada gambar 3.5 menunjukan bahwa *feature* diletakan pada tengah wajah yang miliki nilai matriks sebagai berikut :

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

Matriks tersebut adalah nilai grayscale dari setiap piksel pada citra. Kemudian fitur *haar* yang akan diimplementasikan ialah sebagai berikut :

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

Agar dapat mengetahui nilai fitur *haar* rumus yang digunakan adalah rumus pada persamaan 3.2. Karena itu pertama harus diketahui terlebih dahulu nilai masing-masing bagian fitur. Jika menggunakan penjumlahan untuk menghitung nilai piksel masing-masing fitur yaitu dengan menjumlahkan semuanya seperti pada contoh berikut ini:

Daerah Hitam 1 =

2	1	2	7	4	8
4	8	2	4	1	7

Daerah Putih 1 =

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

Daerah Hitam 2 =

5	8	2	5	6	8
4	5	1	6	2	7

Daerah Hitam 1 = 2 + 1 + 2 + 7 + 4 + 8 + 4 + 8 + 2 + 4 + 1 + 7 = 50

Daerah Hitam 2 = 5 + 8 + 2 + 5 + 6 + 8 + 4 + 5 + 1 + 6 + 2 + 7 = 59

Daerah Putih 1 = 3 + 4 + 6 + 5 + 2 + 7 + 4 + 9 + 5 + 8 + 8 + 6 = 67

NILAI FITUR = |(*total piksel hitam*) – (*total piksel putih*)|

NILAI FITUR = |(50+59) – (67) |

NILAI FITUR = 42

Setelah pencarian nilai fitur proses terakhir yaitu membuat *cascade classifier*. Dapat diketahui *Haar like features* memiliki sifat *learner* dan *classifier* yang lemah. Jadi untuk menambah akurasi perlu dilakukan proses haar-like feature secara massal. Proses *haar like feature* yang dilakukan secara massal dan terorganisir disebut dengan *cascade classifier*. biasanya dilakukan menggunakan stage filter dengan jumlah fitur yang berbeda-beda. Jika nilai fitur tidak memenuhi maka hasil langsung menolak. Contoh sederhana *cascade classifier* adalah sebagai berikut :

1. *Stage* Filter 1 (3 fitur *haar*)
2. *Stage* Filter 2 (5 fitur *haar*)
3. *Stage* Filter 3 (10 fitur *haar*)
4. *Stage* Filter 4 (20 fitur *haar*)

Masing-masing dari *stage* memiliki *threshold* yang berbeda kemudian setiap fitur dalam *stage* juga memilki *threshold* yang berbeda-beda. Objek yang tidak memnuhi akan ditolak dan yang memenuhi akan melalui *stage* selanjutnya hingga objek ditemukan.

Maka setelah mengetahui proses perancangan *face detection* selanjutnya adalah merancang algoritma *face detection* dengan menggunakan *casscade classifier* yang sudah tersedia secara gratis pada internet. Pada tugas akhir ini menggunakan haarcascade_frontalface_dafault.xml yang merupakan *casscade classifier* untuk melakukan pendeteksian wajah yang dibuat oleh open.cv

3.5 *Training Dataset*

Proses selanjutnya adalah mencari pola ciri wajah dari masing-masing individu untuk dapat membedakan antara individu satu dengan yang lainnya. Dibutuhkan operator untuk klasifikasi tekstur dan pada tugas akhir ini menggunakan *Local Binary Patern*. Cara kerjanya adalah dengan mengkonversi seluruh citra *grayscale* yang telah dijadikan *dataset* sebelumnya menjadi *Local Binary Patern* Image dan dijadikan sebuah Histogram. Berikut ini adalah langkah-langkah membuat *Local Binary Patern*.

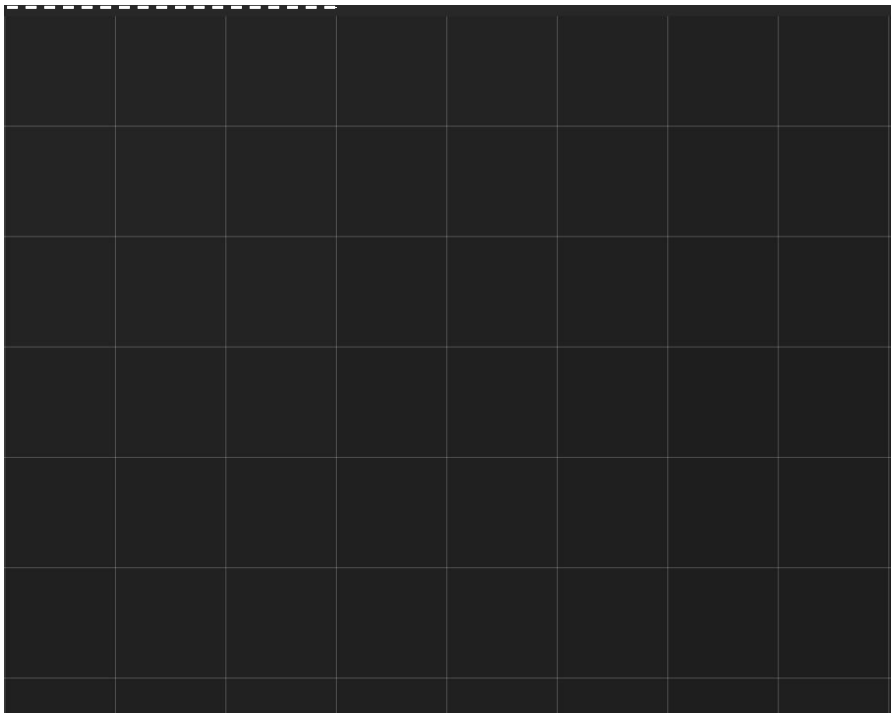
1. Langkah pertama menyiapkan citra *grayscale* yang sudah dijadikan *dataset* sebelumnya lalu akan diubah menjadi *local binary pattern image.* Maka Gambar 3.6 merupakan citra *grayscale*

Gambar 3.6 Citra grayscale

2. Langkah kedua menggunakan rumus LBP 3 x 3 yang dihitung mulai dari piksel pojok kiri atas. Rumus yang akan digunakan diletakan pada persamaan 3.3

$$LBP_{P,R} = \sum_{p=0}^{p-1} S\,(g_p - g_c)\,2^p \quad S = (x) = \begin{cases} 1, if\ x \geq 0; \\ 0, otherwise. \end{cases} \qquad (3.3)$$

3. Setelah dilakukan *zoom* hingga terlihat jelas setiap piksel yang diambil matrik 3 x 3 pada piksel, maka gambar akan menjadi seperti pada gambar 3.7



Gambar 3.7 Matrik 3 x 3 pojok kiri atas

4. Lalu menghitung nilai matrik 3 x 3 yang telah ditandai tersebut dengan menggunakan rumus LBP. Nilai matriknya adalah sebagai berikut:

254	254	255
239	255	254
245	255	249

Titik pusatnya ialah 255 kemudian bandingkan dengan matrik disekitarnya dengan aturan jika pusat ≥ nilai piksel sekitar maka beri nilai 0. Jika titik pusat < nilai piksel sekitar maka beri nilai 1. Sesudah dilakukan seperti itu kemudian matriknya akan menjadi seperti berikut ini :

0	0	1
0	Pusat	0
0	1	0

Untuk mengetahui nilai pusat , hasil biner tersebut kemudian diubah menjadi desimal dengan cara sebagai berikut ini :

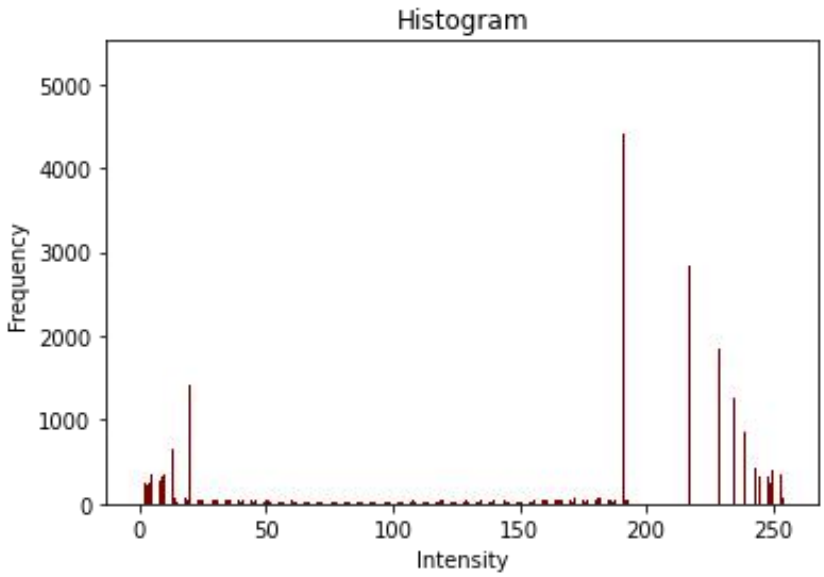
0_7	0_6	1_5
0_0	Pusat	0_4
0_1	1_2	0_3

$$0_7\ 0_6\ 1_5\ 0_3\ 1_2\ 0_1\ 0_0$$
$$2^5 \quad 2^2$$
$$0+0+32+0+0+4+0+0 = 36$$

Dari hasil perhitungan tersebut diketahui nilai pusatnya adalah 36. Lalu akan diimplementasikan perhitungan pada semua bagian citra maka citra LBP akan menjadi seperti pada gambar 3.8.

Gambar 3.8 LBP image

Hasil perhitungan LBP jika ditampilkan dalam bentuk histogram akan menjadi seperti pada gambar 3.9 sebagai berikut :



Gambar 3.9 Local Binary Pattern Histogram

Sesudah menerapkan proses LBP maka histogram dari setiap gambar akan di ekstrak berdasarkan jumlah *grid* (X dan Y) yang dilewatkan parameter. Setelah histogram dari setiap wilayah di ekstrak maka semua histogram yang ada akan digabungkan dan dibuatlah satu histogram baru yang ada digunakan untuk mempresentasikan gambar. Kemudian setiap hasil tersebut akan disimpan dalam sebuah file .yml.

Setelah melalui proses tersebut maka selanjutnya akan membuat algoritma yang dapat menyimpan hasil LBP dalam bentuk yml. Algoritma akan dirancang dalam bentuk *flowchart*. Untuk fungsi *Local Binary Pattern Histogram* menggunakan library buatan Open CV yaitu cv2.createRecognizer. pada *library* tersebut sudah mencakup keseluruhan cara kerja *Local Binary Pattern*. Maka pada gambar 3.7 akan dijelaskan *flowchart* yang akan dibuat.

Gambar 3.7 Flowchart training data menggunakan LBPH

3.6 Face Recognition

Setelah melakukan proses *training* selanjutnya adalah memuat hasil *training* yaitu berupa file .yml yang telah disimpan sebelumnya dan dibandingkan dengan *face detection* maka akan dilakukan. Tahap ini disebut dengan *face recognition* yang artinya pengenalan wajah dimana dataset wajah yang telah disimpan sebelumnya kemudian dibandingan dengan data yang baru. Jika *match* maka wajah akan dapat dikenali sebagai seseorang yang melakukan presensi. Dengan cara menambahkan *variable* nama pada pemilik wajah.