

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pandemi Covid-19 yang saat ini sedang berlangsung telah menjadi perhatian utama masyarakat dunia dan membawa beragam implikasi, baik dalam bidang kesehatan, kebijakan publik, kesejahteraan, pendidikan, sosial, dan lainnya. Khususnya dalam bidang pendidikan membawa dampak terhadap proses pembelajaran. Sejak meluasnya pandemi Covid-19 di dunia, termasuk telah melanda Negara Indonesia, maka sebagai usaha yang dilakukan untuk memutus rantai Covid-19 yaitu tidak boleh adanya kerumunan massa dan melakukan kontak fisik secara langsung baik di bidang apapun.

Dengan perkembangan zaman pada saat ini, pada ilmu Kecerdasan buatan (*Artificial Intelligence*) merupakan salah satu bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti yang dilakukan oleh manusia bahkan bisa lebih baik dari pada yang dilakukan manusia.

Salah satu contohnya sistem presensi, Yaitu merupakan tanda bukti kehadiran seseorang dalam menghadiri suatu pekerjaan atau aktivitas tertentu. Presensi dilakukan dengan cara menginput data seseorang dalam media tertentu sehingga didapatkan laporan presensi. Data yang diinput dalam hal ini bisa bermacam – macam bentuknya. Sebelum era digital datang, absen dilakukan dengan cara menuliskan nama pada selembar kertas atau buku tamu. Penulisan secara manual saat ini masih belum efektif mengingat keakuratan, konsistensi dan kejadian – kejadian yang tidak diinginkan seperti kertas hilang, basah, rusak masih belum maksimal. Demikian juga pada waktu yang digunakan belum maksimal lalu terdapat juga perusahaan yang masih melibatkan kontak fisik untuk melakukan absensi dengan menggunakan sistem *finger print*.

Oleh karena itu dibutuhkan suatu sistem untuk melakukan absensi dengan menggunakan pengenalan wajah seseorang. Sistem pengenalan wajah dapat dibangun dengan menggunakan berbagai macam metode dan algoritma. Salah satu nya dengan algoritma LBPH (*Local Binary Pattern Histogram*) yang digunakan dalam tugas akhir ini.

Algoritma LBPH adalah *Local Binary Pattern Histogram* merupakan salah satu jenis algoritma *machine learning*. LBPH merupakan hasil kombinasi dari dua metode yaitu metode *Local Binary Patterns* (LBP) yang dikombinasikan dengan *Histograms of Oriented Gradients* (HOG) . LBP merupakan operator tekstur yang mempunyai tugas memberi label pixel suatu gambar dengan cara membedakan lingkungan dari setiap *pixel* dan menganggap hasilnya sebagai angka biner sedangkan *Histograms of Oriented Gradient* diperoleh dengan mencari *histogram* dari citra hasil LBP sehingga metode LBPH merupakan metode yang digunakan untuk mendapatkan fitur yang ada pada wajah seseorang berdasarkan histogram yang diperoleh dari hasil LBP (Kosasih, R., & Daomara, C. 2021).

Maka dari itu dibutuhkan suatu sistem untuk melakukan presensi dengan menggunakan pengenalan wajah. Dengan menggunakan salah satu metode algoritma LBPH (*Local Binary Pattern Histogram*).

1.2 Perumusan Masalah

Berdasarkan latar belakang yang disebutkan diatas, masalah utama yang akan dibahas adalah sebagai berikut: Bagaimana cara melakukan implementasi algoritma LBPH (*Local Binary Pattern Histogram*) untuk dapat mengenali wajah *user* agar dapat melakukan pengenalan wajah secara *real time* dan melakukan presensi.

1.3 Tujuan Penelitian

Tujuan dari tugas akhir ini adalah untuk dapat mendeteksi wajah manusia agar dapat melakukan pengenalan wajah secara *real time*. Serta mengurangi kontak fisik agar mencegah penularan virus covid-19 dan juga dapat membantu untuk melakukan presensi lebih cepat dan praktis.

1.4 Batasan Masalah

Agar pengerjaan tugas akhir ini menjadi lebih terarah dan mendapatkan hasil yang lebih spesifik, maka sistem yang dirancang akan dibatasi dalam batasan masalah sebagai berikut:

- Dataset yang digunakan dalam membuat model *machine learning* ini adalah dataset yang telah diambil dari user yang telah didaftarkan.
- Sistem yang dirancang menggunakan bahasa python.
- Algoritma yang digunakan untuk dapat melakukan pengenalan wajah adalah *Local Binary Pattern Histogram* (LBPH)
- Bagian tubuh yang dapat dikenali hanya pada wajah yang sudah didaftarkan saja.
- Sistem tidak dapat mendeteksi suhu tubuh dan penggunaan masker.

1.5 State of The Art

Judul Jurnal	Pembahasan
<p>Implementasi <i>Face Recognition</i> Untuk Mengakses Ruangan</p> <p>Peneliti Alwan Suryansah , Roni Habibi , Rolly Maulana Awangga , Rd. Nuraini Siti Fatonah.</p> <p>Lokasi Politeknik Pos Indonesia</p> <p>Tahun 2020</p> <p>Nama Jurnal</p>	<p><u>Hasil Penelitian :</u></p> <p>Hasil yang didapat pada jurnal ini adalah dapat mengimplementasikan algoritma LBPH (Local Binary Pattern Histogram) untuk pengenalan karakter wajah. memaksimalkan penggunaan komponen-komponen elektronik agar dapat digunakan sebagai alat yang dapat mengenal karakter wajah agar dapat mengakses ruangan.</p> <p><u>Alasan Menjadi Tinjauan Penelitian :</u></p> <p>Dengan pengimplementasian <i>Face Recognition</i> pada jurnal ini dapat dijadikan acuan dalam membangun sistem <i>Face Recognition</i> untuk melakukan presensi</p>

Jurnal Media Pendidikan Teknik Informatika dan Komputer.	dalam mencari metode yang tepat untuk digunakan.
<p>Sistem Pengenalan Wajah dengan <i>Algoritma Haar Cascade</i> dan <i>Local Binary Pattern Histogram</i></p> <p>Peneliti</p> <p>Sayeed Al-Aidid , Daniel S. Pamungkas</p> <p>Lokasi</p> <p>Politeknik Negeri Batam</p> <p>Tahun</p> <p>2018</p> <p>Nama Jurnal :</p> <p>Jurnal Rekayasa Elektrika</p>	<p><u>Hasil Penelitian :</u></p> <p>Pada jurnal ini membahas tentang pembuatan Aplikasi pengenalan wajah dengan memanfaatkan 2 algoritma dan menggabungkan 2 metode yaitu <i>Haar Cascade</i> dengan algoritma <i>Local Binary Pattern Histogram</i> dapat digunakan sebagai pendeteksi dan pengenalan wajah manusia.</p> <p><u>Alasan Menjadi Tinjauan Penelitian :</u></p> <p>Dengan melihat penggunaan kedua algoritma tersebut yaitu algoritma <i>Haar Cascade</i> dengan algoritma <i>Local Binary Pattern Histogram</i> dapat melakukan pendeteksian wajah dan pengenalan wajah yang cukup baik dan mudah dimengerti , sehingga kedua algoritma tersebut dapat digunakan sebagai acuan untuk membuat sistem <i>Face Recognition</i> untuk melakukan presensi.</p>

<p>Face Recognition Sebagai Sistem Pendataan dan Akses Masuk Perpustakaan Daerah</p> <p>Peneliti</p> <p>Risyaf Fawwaz Pradipta , Denny Darlis , Syahban Rangkuti</p> <p>Lokasi</p> <p>Universitas Telkom</p> <p>Tahun</p> <p>2020</p> <p>Nama Jurnal</p> <p>Jurnal Teknologi Telekomunikasi</p>	<p><u>Hasil Penelitian :</u></p> <p>Jurnal ini membahas tentang Sistem aplikasi <i>Face recognition</i> yang digunakan untuk pendataan dan akses masuk perpustakaan daerah dapat berjalan secara baik. Lalu memanfaatkan <i>face detection</i> dengan metode <i>Viola-Jones</i>. dan juga memanfaatkan salah satu <i>domain</i> kecerdasan buatan yaitu CV (<i>Computer Visual</i>).</p> <p><u>Alasan Menjadi Tinjauan Penelitian :</u></p> <p>Karena jurnal ini dapat memanfaatkan salah satu <i>domain</i> kecerdasan buatan yaitu CV (<i>Computer Visual</i>) dengan memanfaatkan <i>face detection</i> dengan menggunakan metode <i>Viola-Jones</i>. Jurnal ini dapat menjadi acuan untuk pembuatan Sistem <i>Face Recognition</i> untuk melakukan presensi.</p>
<p>Pengenalan Wajah dengan Menggunakan Metode <i>Local Binary Patterns Histograms</i> (LBPH)</p> <p>Peneliti</p> <p>Rifki Kosasih , Christian Daomara</p> <p>Lokasi</p> <p>Universitas Gunadarma, Depok, Indonesia</p>	<p><u>Hasil Penelitian :</u></p> <p>Pada jurnal Ini membahas penelitian tentang absensi otomatis untuk membedakan wajah orang ke-1 dengan orang lainnya. Metode yang digunakan untuk melakukan pengenalan wajah secara otomatis. Pada penelitian ini digunakan metode <i>Local Binary Patterns Histograms</i> (LBPH) yang merupakan kombinasi dari metode Local Binary Patterns (LBP).</p>

<p>Tahun</p> <p>2021</p> <p>Nama Jurnal</p> <p>Jurnal Media Informatika Budidarma</p>	<p><u>Alasan Menjadi Tinjauan Penelitian :</u></p> <p>Karena langkah-langkah yang digunakan pada metode <i>Local Binary Pattern Histogram</i> (LBPH) dijelaskan secara rinci dan mudah dimengerti sehingga dapat dijadikan acuan untuk membuat sistem <i>Face Recognition</i> untuk melakukan presensi.</p>
<p>Implementasi Face Recognition pada Absensi Kehadiran Mahasiswa Menggunakan Metode Haar Cascade Classifier</p> <p>Peneliti</p> <p>Munawir, Liza Fitria, Muhammad Hermansyah</p> <p>Lokasi</p> <p>Universitas Samudra</p> <p>Tahun</p> <p>2020</p> <p>Nama Jurnal</p> <p>Jurnal Nasional Informatika dan Teknologi Jaringan</p>	<p><u>Hasil Penelitian :</u></p> <p>Pada jurnal ini meneliti dan melakukan pengujian Sistem pengenalan dengan wajah dapat diterapkan pada absensi kehadiran mahasiswa. Dengan menggunakan metode <i>Haar Cascade Classifier</i>. Tetapi Sistem pengenalan wajah dengan banyak wajah (<i>multiple face recognition</i>) kurang cocok untuk diterapkan pada absensi kehadiran mahasiswa, karena terdapat banyak kesalahan dalam mengenali wajah sehingga proses absensi tidak sesuai dengan data yang sebenarnya. Dengan hasil pengujian satu wajah adalah 76% dan pengujian banyak wajah adalah 33.33%.</p> <p><u>Alasan Menjadi Tinjauan Penelitian :</u></p> <p>Dengan melihat penggunaan algoritma <i>Haar Cascade Classifier</i> yang dapat melakukan absensi kehadiran mahasiswa dengan cukup baik maka algoritma <i>Haar Cascade Classifier</i> dapat</p>

	diimplementasikan untuk pembuatan sistem Face Recognition untuk melakukan presensi.
--	---

1.6 Sistematika Penulisan

Sistematika penulisan disusun untuk memberikan gambaran secara umum mengenai permasalahan dan pemecahannya. Penyusunan ini diuraikan dalam beberapa pokok permasalahan yang terbagi dalam beberapa bab. Sistematika penulisan Tugas Akhir ini adalah sebagai berikut:

Bab 1 PENDAHULUAN

Bab ini memuat pendahuluan penelitian yang terdiri dari latar belakang, rumusan masalah, tujuan dan manfaat penelitian, ruang lingkup penelitian, *state of the art*, serta sistematika penulisan penelitian.

Bab 2 TINJUAN PUSTAKA

Bab ini memuat tinjauan pustaka penelitian yang terdiri dari teori dasar mengenai sistem *Computer Vision* (CV), *Artificial intelligence* (AI), *Machine Learning*, Algoritma *Haar Cascade Classifier*, Algoritma *Local Binary Pattern Histogram* (LBPH), *OpenCV*.

Bab 3 METODE DAN PERANCANGAN

Bab ini berisi tentang metode dan perancangan sebuah sistem yang dapat melakukan pengenalan wajah dan melakukan presensi. Dengan menggunakan algoritma LBPH (*Local Binary Pattern Histogram*).

Bab 4 PEMBAHASAN

Bab ini membahas mengenai sistem yang akan dijalankan secara terstruktur menggunakan algoritma LBPH (*Local Binary Pattern Histogram*). Untuk hasil akhirnya akan berupa pengujian dan analisa terhadap pengenalan wajah.

Bab 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran dari pembahasan sistem Pengenalan Wajah yang telah dibangun dengan menggunakan algoritma LBPH (*Local Binary Pattern Histogram*).

DAFTAR REFRENSI**LAMPIRAN**

BAB 2

TINJAUAN PUSTAKA

2.1 Face Recognition

Face recognition adalah teknik biometrik yang memungkinkan komputer atau mesin untuk mengenali wajah manusia. Teknologi pengenalan wajah dapat diterapkan dalam kehidupan sehari-hari untuk mempermudah aktivitas manusia. Dalam implementasi pengenalan wajah, banyak metode yang dapat digunakan, salah satunya adalah *machine learning*, yang mana hasil tangkapan dari kamera akan dilakukan pencocokan dengan menggunakan data yang telah dilatih sebelumnya menggunakan teknologi *machine learning*. Untuk dapat membaca dan memproses data inputan berupa citra, dapat menggunakan *library* OpenCV (Prapdipta, Darlis dan Rangkuti. 2020).

2.2 Computer Vision

Computer vision merupakan cabang dari *artificial intelligence* (AI) atau kecerdasan buatan yang mempelajari ilmu mengenai bagaimana personal komputer bisa mengenali objek yang diamati. Implementasi dari teknologi AI sudah banyak sekali digunakan, baik pada teknologi *smartphone* juga pada dunia robotika. *computer vision* juga memungkinkan komputer bisa melihat objek atau benda yang terdapat pada lingkungan sekitar. Maka komputer bisa menganalisis benda atau gambar yang terdapat pada depannya sehingga informasi yang sudah terdeteksi bisa diterima & mampu membentuk perintah tertentu (Santoso, B. 2021).

2.2.1 OpenCv

Library OpenCV (*Open Capture Vision*) merupakan suatu pemrograman aplikasi yang memungkinkan komputer untuk menampilkan objek seperti wajah dan objek lainnya sehingga digunakan untuk pemrosesan citra atau yang disebut dengan *image processing* secara *real-time*. *Library* ini biasa ditujukan untuk *computer vision*, lalu komputer dapat mengambil keputusan, mengambil tindakan, dan mengenali objek yang sarasannya dalam penelitian ini adalah wajah.



Gambar 2. 1 Logo OpenCV

Library OpenCV saat ini bersifat *open source* atau gratis untuk digunakan oleh siapa saja dengan berbagai algoritma *computer vision* (Warnilah, A. I., Jaya-Mulyana, A., Siti-Nuraeni, F., & Aninditya-Widianto, T. 2022).

2.3 Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan buatan atau biasa disebut juga dengan *artificial intelligence* (AI) merupakan teknologi di bidang ilmu komputer yang menjelaskan tentang kecerdasan manusia ke dalam mesin (komputer) untuk menyelesaikan berbagai persoalan dan pekerjaan seperti yang dilakukan oleh manusia bahkan lebih baik dari manusia. Kecerdasan buatan adalah kecerdasan yang ditambahkan oleh manusia ke dalam suatu sistem teknologi, diatur dan dikembangkan dalam konteks ilmiah, bentukan dari kecerdasan entitas ilmiah yang ada.

Pada dasarnya kecerdasan buatan atau *artificial intelligence* (AI) adalah suatu pengetahuan yang membuat komputer dapat meniru kecerdasan manusia sehingga komputer dapat melakukan hal-hal yang dikerjakan oleh manusia di mana membutuhkan suatu kecerdasan, contohnya melakukan analisis penalaran untuk mengambil kesimpulan atau keputusan. (Subakti, H., Romli, I., Nur Syamsiyah, S. T., Budiman, A. A., Kom, M., Herianto, S. P., & MSI, M. 2022).

2.4 *Machine Learning*

Machine Learning merupakan bidang studi yang berfokus pada desain dan analisis algoritma untuk melakukan komputer dapat belajar tanpa diprogram secara eksplisit. *Machine Learning* berisi algoritma umum yang bersifat umum dimana algoritma tersebut dapat menghasilkan hal-hal yang berguna dari kumpulan data tertentu tanpa harus menulis kode yang spesifik. Pada intinya adalah algoritma yang umum tersebut ketika diberikan sejumlah data maka akan dibangun sebuah sistem

atau model yang berasal dari data tersebut. Sebagai contohnya adalah sebuah algoritma untuk mengenali tulisan tangan dapat digunakan untuk dapat mendeteksi email yang berisi spam dan bukan spam tanpa harus mengganti kode. Algoritma yang sama ketika diberikan data pelatihan yang berbeda akan menghasilkan logika klasifikasi yang berbeda. Berdasarkan cara belajar *machine learning* agar berfungsi, *machine learning* dapat dikategorikan pembagian pendekatan *machine learning* dipisahkan menjadi tiga kategori, Yaitu : (MACHINE LEARNING : Teori, Studi Kasus dan Implementasi Menggunakan Python, 2021).

2.4.1. Supervised Learning

Supervised learning adalah metode pembelajaran klasifikasi yang terarah dengan proses pembelajaran dimana seluruh kumpulan data akan diberikan label. Dalam *Supervised learning* model dapat dilatih dalam dataset *training* dan diawasi untuk membuat klasifikasi atau prediksi sesuai dengan *output* berupa data berlabel yang ditentukan sebelumnya berdasarkan pola data *training* yang ada (Hussein Saddam, 2022).

2.4.2 Unsupervised Learning

Unsupervised Learning merupakan kebalikannya dimana proses pembelajaran dapat dilakukan tanpa adanya petunjuk atau arahan. Algoritma pada komputer yang bekerja untuk menemukan pola di dalam data secara matematis, metode *unsupervised learning* terjadi ketika memiliki sejumlah data masukan (X) dan tanpa variabel *output* yang berhubungan. Dengan pembelajaran yang menggunakan algoritma machine learning ini dapat melakukan analisa dan mengklasifikasikan kumpulan data yang tidak berlabel. Algoritma ini disebut tanpa pengawasan karena menemukan pola tersembunyi dalam data tanpa perlu campur tangan manusia (Hussein Saddam, 2022).

2.4.3. Reinforcement Learning

Reinforcement learning merupakan sebuah komputer yang akan berinteraksi dengan sebuah lingkungan yang sangat dinamis dimana komputer dapat melakukan tugas tertentu. Melalui sebuah algoritma, mesin akan mempelajari bagaimana cara membuat keputusan yang spesifik berdasarkan lingkungan yang berubah-ubah. Selain itu cara belajar tersebut, *machine*

learning dapat juga dikategorikan berdasarkan proses pembelajaran yang akan dilakukan secara bertahap (*Batch Learning*) atau secara langsung (*on the fly – Online Learning*). Dasar dari *online learning* dapat menghasilkan suatu model yang dapat melakukan proses pembelajaran data baru secara *realtime* atau mendekati *realtime*. Kemudian sedangkan *Batch Learning*, *data training* akan di bagi-bagi menjadi beberapa bagian lalu setiap bagiannya akan dipelajari secara terpisah pada waktu yang berbeda (MACHINE LEARNING : Teori, Studi Kasus dan Implementasi Menggunakan Python, 2021).

2.5 Pengolahan Citra Digital

Pengolahan citra digital secara umum menunjuk pada proses komputer pada citra dua dimensi. Bahkan disebutkan Anil K Jain (Jain, 1989) dalam pengertian yang luas pengolahan citra digital merupakan implementasi dari suatu proses digital pada data dua dimensi. Dalam sebuah citra yang mengalami penurunan kualitas karena mengandung *noise*, *blurring*, kontras, kurang tajam, buruk, dan lainnya akan sulit di interpretasikan baik oleh manusia ataupun oleh komputer. Sehingga pada citra itu perlu dilakukan sejumlah pemrosesan dan manipulasi agar dapat menghasilkan citra dengan kualitas informasi yang lebih baik, sehingga dapat mudah di interpretasi.

Operasi pengolahan citra diterapkan pada citra bila:

- Modifikasi citra perlu dilakukan agar meningkatkan kualitas penampakan atau untuk melihatkan beberapa aspek informasi yang ada dalam citra (*image enhancement*). Operasi ini memiliki tujuan memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Contohnya dapat dilakukan dengan perbaikan kontras, penajaman dan lainnya.
- Terdapat cacat pada citra sehingga perlu dihilangkan (*image retoration*). Contohnya dapat dilakukan dengan cara penghapusan *noise*, *debluring* (penghilangan kesamaran) karena citra terlihat kabur.
- Elemen dalam citra harus dikelompokkan, dicocokkan dan diukur (*segmentation*). Operasi ini berkaitan erat dengan (*pattern recognition*) pengenalan pola.
- Perlu ekstraksi ciri-ciri tertentu (contoh: tekstur, warna, bentuk) dalam pengidentifikasian objek (*image analysis*).

- Sebagian citra perlu digabung dengan citra lain (image reconstruction). contohnya adalah beberapa foto rontgen digunakan untuk membentuk ulang gambar organ tubuh.
- Menyembunyikan informasi rahasia (berupa teks/citra) dalam citra (*steganografi*).

-

Aspek dalam pengolahan citra digital meliputi peningkatan kualitas citra, restorasi citra, segmentasi citra. Secara bururutan peningkatan kualitas citra. Yang dilakukan dalam pengolahan citra digital yaitu:

- Akuisisi

Proses akuisisi citra adalah pemetaan suatu pandangan menjadi citra kontinu dengan menggunakan sensor. Contohnya kita dapat membuat sebuah citra digital dengan kamera atau scanner.

- Preprocessing

Pada praproses ini bertujuan untuk meningkatkan kualitas kecerahan dan kontras, penajaman suatu citra, menghilangkan *noise*, menghilangkan gangguan geometrik/radiometrik.

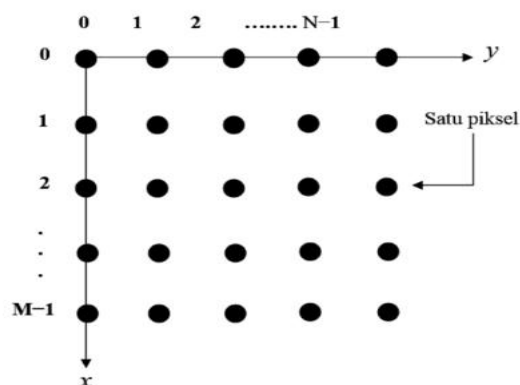
- Segmentasi

Melakukan partisi citra menjadi *region of interest* (ROI) atau wilayah- wilayah objek (*internal properties*), menentukan garis batas wilayah objek (*external shape characteristics*).

Definisi citra adalah sesuatu yang merepresentasikan objek atau beberapa objek dalam bidang dua dimensi. Atau juga bisa dikatakan citra adalah visualisasi, kemiripan dari suatu objek. Citra dapat dikategorikan kedalam citra kontinu dan citra digital. Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog. Citra diksrit/digital dihasilkan melalui proses digitalisasi terhadap citra kontinu. Suatu citra didefinisikan sebagai $f(x,y)$ memiliki ukuran M baris dan N kolom dengan x dan y merupakan koordinat spasial, dan amplitude f dititik koordinat (x,y) sebagai intensitas (skala keabuan). Disebut sebagai citra digital jika nilai (x,y) dan nilai amplitude f secara menyeluruh (*finite*) dan bernilai diskrit.

Citra digital adalah citra yang dapat diolah dengan komputer. Lalu sedangkan citra dihasilkan dari peralatan digital (citra digital) langsung dapat diolah dengan komputer. Karena penyebabnya adalah di dalam peralatan digital terdapat sistem sampling dan kuantisasi. Sedangkan peralatan analog tidak dilengkapi kedua sistem tersebut. Sistem sampling merupakan sistem yang dapat mengubah citra kontinu menjadi citra digital dengan cara membagi citra analog menjadi M baris dan N kolom, sehingga menjadi citra diskrit. Semakin besar nilai M dan N, semakin halus juga citra digital yang akan dihasilkan. Pertemuan antara baris dan kolom itu disebut dengan piksel. Sistem kuantisasi adalah sistem yang melakukan pengubahan intensitas analog ke intensitas diskrit, sehingga dengan proses tersebut dilakukan untuk membuat gradasi warna sesuai dengan kebutuhan. Maka kedua sistem inilah yang bertugas untuk memotong-motong citra menjadi M baris dan N kolom (proses sampling) sekaligus menentukan besarnya intensitas yang terdapat pada titik tersebut (proses kuantisasi), sehingga dapat menghasilkan resolusi citra yang akan diinginkan.

Hasil sampling dan kuantisasi dari sebuah citra adalah bilangan real yang membentuk sebuah matriks M baris dan N kolom. Maka ukuran citra adalah $M \times N$. Secara umum, sistem koordinat yang dipergunakan untuk mewakili citra dalam teori pengolahan citra seperti digambarkan pada gambar 2.1. Yaitu seperti citra digital diwakili oleh matriks yang terdiri dari M baris dan N kolom, di mana perpotongan antara baris dan kolom disebut piksel. Maka piksel memiliki dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang ada pada koordinat (x,y) adalah $f(x,y)$, yaitu besar intensitas atau warna dari piksel di titik itu.



Gambar 2. 2 Sistem koordinat yang digunakan mewakili citra

Maka, Sebuah citra digital dapat juga ditulis dalam bentuk matriks berikut ini pada gambar 2.2 yaitu :

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Gambar 2. 3 Matriks Citra Digital $f(x,y)$

Keterangan :

M = Jumlah *pixel* baris (*row*)

N = Jumlah *pixel* kolom (*column*)

Berdasarkan gambar 2.1, maka secara matematis citra digital dapat dituliskan sebagai fungsi intensitas $f(x,y)$, dimana harga x (baris) dan y (kolom) merupakan koordinat posisi dan $f(x,y)$ merupakan nilai fungsi pada setiap titik (x,y) yang mempunyai besar intensitas citra atau tingkat keabuan atau warna dari piksel titik tersebut (Zonyfar, C. 2020).

2.5.1 Jenis Citra Digital

Secara umum, citra digital dibagi menjadi tiga jenis: citra berskala keabuan, citra biner dan citra berwarna.

a. Citra Berskala Keabuan

Citra berskala keabuan atau biasa disebut dengan citra *grayscale* merupakan citra yang hanya mempunyai jumlah kanal satu yang ditampilkan hanyalah nilai intensitasnya saja. Satu kanal setiap piksel berwarna abu-abu, biasanya dari 0 (hitam) hingga 255 (putih). Dalam rentang tersebut setiap piksel diwakili oleh 1 byte (8 bit). Sehingga citra ini sering juga dikenal sebagai citra 8 bit, citra *intensity*, *grayscale*, atau *gray level*. Cara mengubah citra berwarna RGB kedalam citra berskala keabuan dapat dilakukan dengan cara merata-ratakan semua nilai piksel pada RGB. Seperti pada persamaan sebagai berikut ini :

$$y = \frac{1}{3}(R + G + B) \quad (2.1)$$

Sedangkan pada MATLAB untuk dapat mengubah citra berwarna menjadi RGB kedalam warna grayscale sesuai dengan persamaan berikut :

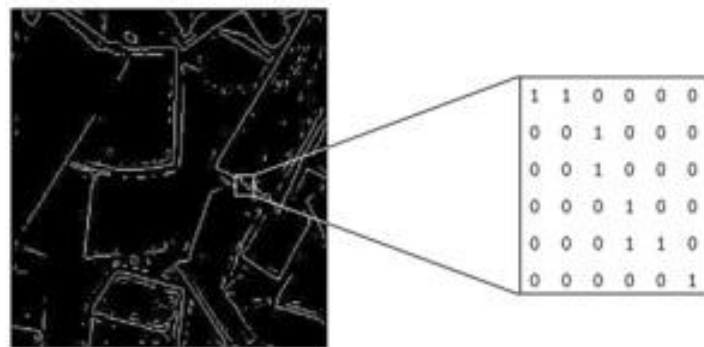
$$Grayscale = 0.299R + 0.587G + 0.144B \quad (2.2)$$



Gambar 2. 4 Citra berskala keabuan

Citra Biner

Citra biner adalah citra yang mana setiap pikselnya hanya mempunyai hitam atau putih. Citra biner merupakan citra yang efisien dalam penyimpanan, dikarenakan hanya membutuhkan satu bit saja per piksel. Jenis citra ini sesuai untuk merepresentasikan teks (cetak atau tulis tangan). Citra biner juga sering berfungsi sebagai *masking* atau proses segmentasi. Agar dapatkan citra biner membutuhkan citra berskala keabuan yang kemudian dilakukan *thresholding*. Apabila nilai piksel pada citra berskala keabuan lebih besar atau sama dengan nilai ambang batas, maka piksel tersebut dapat di ubah menjadi 1.



Gambar 2. 5 Binary Image

b. Citra Berwarna

Citra berwarna merupakan citra yang memiliki 3 kanal warna didalamnya. Citra berwarna (*truecolor*) juga sering disebut dengan citra RGB, karena dimodelkan kedalam ruang warna dengan R (*red*/merah), G (*green*/hijau), B (*blue*/biru) sebagai pembentuk komponennya. RGB terdapat standar yang digunakan untuk menampilkan citra yang berwarna pada layer komputer, walaupun seperti itu terdapat juga citra berwarna yang menggunakan warna lainnya, seperti ruang warna HSV, CMYK, dan Lab (Zonyfar, C. 2020).



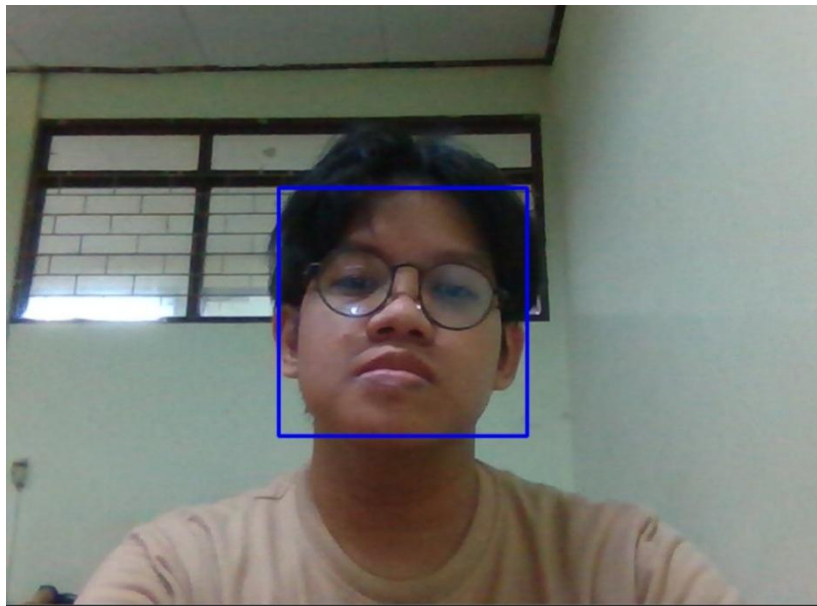
Gambar 2. 6 RGB Image

2.6 Haar Cascade Classifier

Algoritma *Haar Cascade Classifier* merupakan salah satu *library* yang tersedia dalam OpenCV, dibangun dengan menggunakan bahasa C/C++ dengan *API (Application Programming Interface)* python yang dapat digunakan untuk mendeteksi objek pada image digital. Algoritma ini dapat dengan cepat mengenali objek, termasuk wajah manusia, secara *real time*. Algoritma klasifikasi *Haar Cascade* memiliki keunggulan lebih cepat karena hanya bergantung pada jumlah piksel kuadrat pada citra (Abidin, S, 2018).

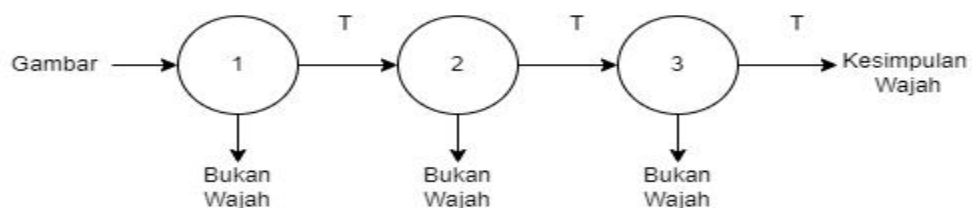
Untuk dapat memproses pendeteksian wajah digunakan algoritma *haar cascade classifier* digunakan untuk mengidentifikasi objek dalam *image digital*. Istilah *haar* berarti menunjukkan suatu fungsi matematika (*Haar Wavelet*) yang berbentuk kotak. Awalnya, pemrosesan gambar hanya terdiri dari pengecekan nilai RGB setiap piksel, tetapi metode ini terbukti tidak efektif. Kemudian Viola dan Jones mengembangkannya menjadi fitur *haar-like feature*. Fungsi *haar-like feature* dapat memproses gambar dalam kotak-kotak yang memiliki banyak piksel di dalam kotak tersebut. kemudian kotak tersebut diproses dan dapat menghasilkan nilai yang berbeda yang menunjukkan area putih dan hitam. Sehingga nilai-nilai ini digunakan sebagai dasar untuk pemrosesan gambar.

Haar like feature memproses citra dalam sebuah kotak persegi dengan menggunakan ukuran tertentu misalnya seperti pada gambar 2.7



Gambar 2. 7 Fitur pada Haar-Like Feature

Pada kotak biru pada gambar 2.7 tersebut terjadi proses *filtering* atau penyaringan untuk menentukan ada atau tidaknya suatu objek yang akan dideteksi. Proses tersebut akan dilakukan secara kompleks dan bertingkat. Seperti yang akan ditunjukkan pada gambar filter berikut ini



Gambar 2. 8 Alur metode haar cascade classifier

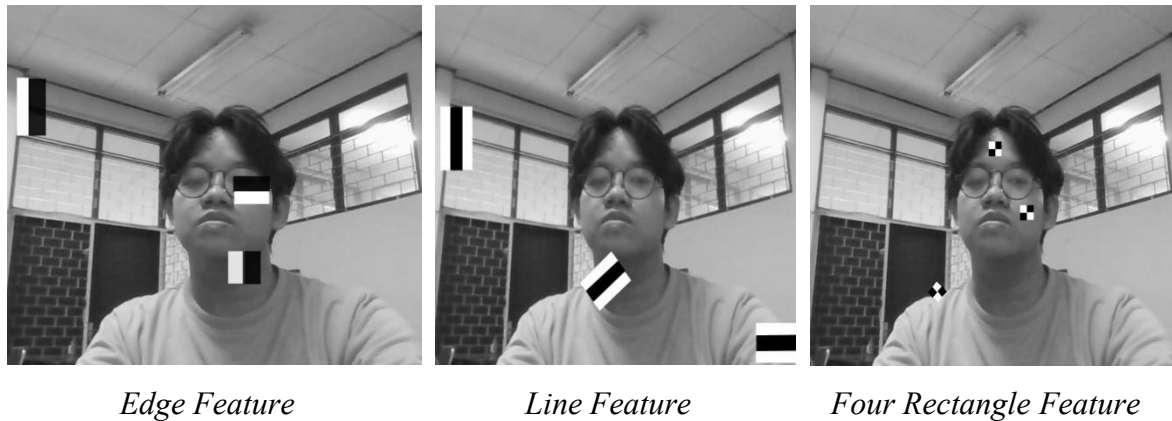
2.6.1 Haar Feature

Haar Feature merupakan fitur yang mempunyai dasar pada *Wavelet Haar*. *Wavelet Haar* adalah gelombang bujur sangkar (1 interval rendah dan interval tinggi). Untuk dua dimensi, yaitu berisi hitam dan putih. Proses pertama yang dilakukan oleh metode *Haar Cascade Classifier* adalah dengan merubah image RGB menjadi citra *grayscale*. Berikut adalah contoh haar-like feature dapat dilihat pada gambar 2.9.



Gambar 2. 9 Contoh perubahan citra RGB menjadi citra Grayscale

Setelah citra *image* dikonversi menjadi citra *grayscale*, proses selanjutnya yaitu memilih fitur Haar yang ada dalam algoritma *Haar Cascade Classifier* disebut dengan *haar-like feature*. Dalam algoritma *Haar Cascade Classifier* Ada beberapa jenis fitur yang bisa digunakan seperti *Edge-feature*, *Line feature*, dan *Four-Rectangle feature*. Pada fitur tersebut digunakan untuk mencari fitur wajah seperti mata, hidung dan mulut. Pada setiap kotak-kotak fitur tersebut terdiri dari beberapa piksel dan akan dihitung selisih antara nilai piksel pada kotak putih dengan nilai piksel pada kotak hitam. Maka apabila nilai selisih antara daerah putih dengan daerah hitam di atas nilai ambang (*threshold*), maka daerah tersebut dinyatakan memiliki fitur. Proses pemilihan fitur dapat dilihat pada gambar 2.10.



Gambar 2. 10 Macam-macam variasi feature pada haar

Untuk dapat memilih fitur mata, hidung, dan mulut maka digunakan kotak-kotak fitur yang bisa dilihat pada gambar 2.11 berikut ini.



Gambar 2. 11 Fitur pada mulut, mata dan hidung.

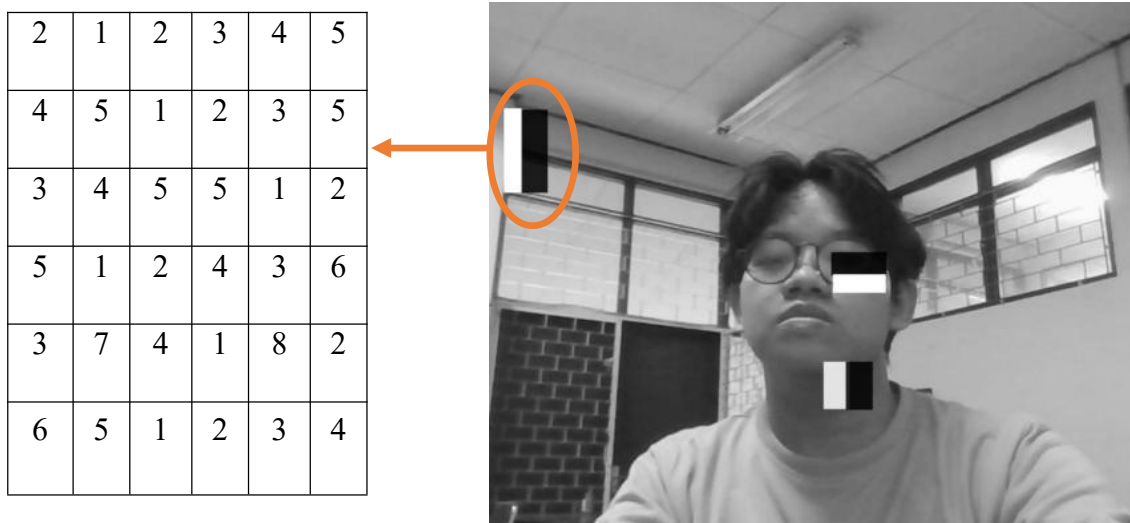
Terdapat fitur *Haar* ditentukan dengan mengurangi rata-rata piksel pada daerah hitam dari rata-rata daerah yang putih. Jika terdapat nilai perbedaannya diatas nilai ambang atau *threshold*, maka dapat dikatakan bahwa terdapat fitur tersebut . Nilai *Haar-like feature* adalah perbedaan dari jumlah nilai-nilai piksel *gray level* dalam kotak hitam dan kotak putih.

Pada umumnya citra wajah yang menghadap ke depan daerah mata, tepi hidung, mulut dan dagu cenderung lebih gelap dibandingkan dengan daerah kedua pipi, dagu dan kening untuk mempermudah dan mempercepat proses perhitungan nilai *Haar* pada sebuah *image*, *Algoritma Haar cascade Classifier* memiliki sebuah perhitungan yang disebut dengan *integral image* (Prathivi, R., & Kurniawati, Y. 2020).

2.6.2 Integral Image

Integral Image merupakan suatu teknik penjumlahan intensitas piksel pada suatu daerah yang menghasilkan representasi baru dari citra sebelumnya. *Integral image* sering digunakan pada algoritma untuk pendeteksian wajah dengan menggunakan *integral image* proses perhitungan dapat dilakukan hanya dengan satu kali scan saja dan membutuhkan waktu yang sangat cepat dan akurat. *Integral image* digunakan untuk menghitung hasil penjumlahan nilai piksel pada daerah yang dideteksi oleh fitur *haar*.

Nilai-nilai piksel yang akan dihitung merupakan nilai-nilai piksel dari sebuah citra masukan yang dilalui oleh fitur *haar* pada saat pencarian fitur wajah. Setiap jenis fitur yang akan digunakan pada setiap kotak-kotaknya terdiri dari beberapa piksel. Jadi apabila ada sebuah citra masukan yang dilalui oleh fitur *haar* dapat dilihat pada gambar 3.12



Gambar 2. 12 Nilai piksel pada sebuah fitur

Dapat dilihat pada gambar 2.12 terdapat sebuah feature haar pada gambar yang telah dikonversi ke *grayscale*, maka untuk dapat menentukan nilai fitur tersebut dapat digunakan pada persamaan rumus 2.3 sebagai berikut.

$$\text{NILAI FITUR} = |(\text{total piksel hitam}) - (\text{total piksel putih})| \quad (2.3)$$

Ada berbagai cara untuk mengetahui nilai dari piksel tersebut, yaitu bisa dilihat pada persamaan rumus 2.2. dan juga dibutuhkan teknik yang dapat dilakukan dengan cepat sehingga dapat diimplementasikan pada ratusan fitur *haar* dengan skala yang berbeda. Teknik yang efisien untuk melakukan itu yaitu *integral image*. Maka perhitungan tersebut dapat ditulis dengan rumus *integral image* sebagai berikut.

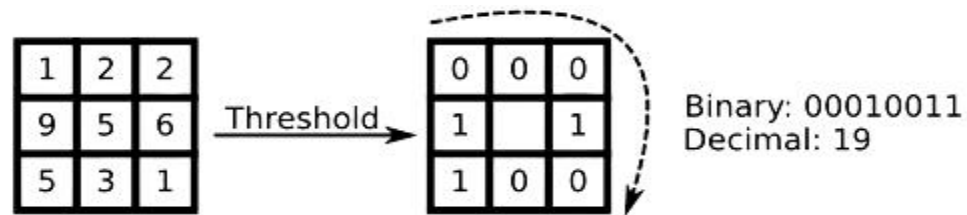
$$s(x, y) = i(x, y) + s(x - 1, y) + s(x, y - 1) - s(x - 1, y - 1) \quad (2.4)$$

Keterangan :

- $s(x, y)$: Merupakan nilai citra integral pada bidang (x, y)
- $i(x, y)$: Intensitas/nilai piksel citra asli.
- $s(x-1, y)$: Merupakan nilai citra integral dari piksel tetangga sebelah kiri.
- $s(x, y-1)$: Merupakan nilai citra integral dari piksel tetangga sebelah atas.
- $s(x-1, y-1)$: Merupakan nilai citra integral diagonal dari $s(x, y)$

2.7 Local Binary Pattern Histogram (LBPH)

Local Binary Pattern (LBP) adalah salah satu dari metode yang terkenal dalam mengenali sebuah objek. Sederhana tetapi sangat efisien yang dapat melabeli *pixel* berdasarkan suatu gambar dengan menggunakan ambang batas lingkungan setiap *pixel* dan hasilnya sebagai bilangan biner. Dengan menggabungkan LBP dan *Histogram* bisa didapatkan deskripsi fitur yang bisa dipakai untuk merepresentasikan gambar wajah. Salah satu metode pendeteksian objek yang paling umum, Dalam hal ini, metode yang digunakan adalah membedakan objek dengan background. *Local Binary Pattern Histogram* (LBPH) merupakan algoritma kombinasi dari LBP dan *Histogram of Oriented Gradients* (HOG). Pengenalan wajah merupakan pengenalan wajah tingkat tinggi, dalam melakukan pengenalan wajah dapat menggunakan pencocokan dengan LBPH. Gambar wajah yang diambil secara *real time* oleh kamera dibandingkan dan dicocokkan dengan histogram yang diambil dari gambar wajah di *database* (Wibowo, A. W., Karima, A., Wiktasari, A. Y., & Fahriah, S., 2020).



Gambar 2. 13 Konversi biner ke desimal

Dari gambar 2.13 terdapat citra yang telah dikonversi dalam bentuk *grayscale*. Citra tersebut diambil sebagian pikselnya sebesar 3 x 3 piksel yang setiap pikselnya memiliki nilai masing-masing. Perbandingan yang dilakukan menggunakan rumus di bawah ini :

$$LBP = \sum_n^c = S(i_n - i_c) 2^n \quad (2.5)$$

(LBP) = Nilai *Local Binary Pattern*

i_c = Nilai piksel pusat

i_n = Nilai tetangga piksel

2.8 Tensorflow

Tensorflow merupakan suatu *framework* yang sudah dikembangkan oleh Google Brain Team pada saat tahun 2015. Saat awal framework tensorflow ini dikembangkan digunakan untuk perhitungan numerik. Dengan semakin berkembangnya teknologi, dalam waktu ini framework tensorflow biasa dipakai untuk pengembangan aplikasi *Artificial Intelligence* (AI) oleh perusahaan-perusahaan besar. Aplikasi yang telah dikembangkan misalnya seperti pengklasifikasi gambar, penyematan kata, & pengembangan *chatbot* (Wiranda, Purba, Sukmawati, 2020).



Gambar 2. 15 Logo Tensorflow

Framework tensorflow kini telah menyediakan interface yang dapat digunakan untuk mengimplementasikan algoritma *machine learning* dan suatu aplikasi yang digunakan untuk menjalankan algoritma. Algoritma pemodelan yang didukung oleh *framework* ada begitu banyak tensorflow, contohnya adalah *Recurrent Neural Network* (RNN), *Local Binary Pattern Histogram* (LBPH), dan eksekusi paralel (Wiranda , Purba, Sukmawati, 2020).

2.9 Python

Python adalah bahasa yang sangat terkenal di antara pengembang *machine learning*, *Data scientists* maupun *Data Miner*. Python juga bahasa pemrograman yang interpretatif multifungsi. Python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintak, dan juga python merupakan pemrograman yang bersifat *open source* dan *multiplatform*, ada beberapa *feature* yang dimiliki Python. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain. Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. hampir semua distronya sudah menyertakan Python di dalamnya. Python digunakan sebagai bahasa yang diimplementasi karena kemudahannya. Python lebih banyak diminati karena ia termasuk bahasa pemrograman yang umum sehingga pengembang dapat membuat aplikasi untuk Enterprise lebih mudah (Id, I. D. 2021).

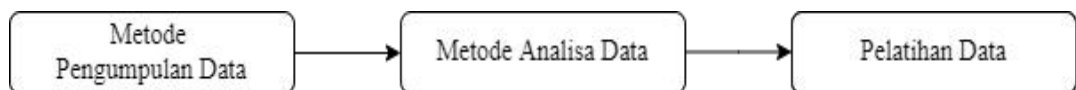


Gambar 2. 16 Logo Pyhton

BAB 3

METODE DAN IMPLEMENTASI

Pada Bagian ini Bab ini akan membahas tentang metode dan implementasi dalam sistem pengenalan wajah untuk melakukan kehadiran otomatis menggunakan algoritma *Local Binary Pattern Histogram* (LBPH). dalam Bab ini terdapat beberapa tahapan yang dalam pembuatan sistem *face recognition* yaitu analisa data yang akan digunakan, lalu tahap-tahap selanjutnya adalah sebagai berikut. Tahapan yang pertama adalah analisa data yang akan digunakan dengan beberapa tahapan dari *preprocessing* untuk mengolah citra yang akan digunakan, Melakukan implementasi metode *Haar Cascade Classifier* dan *Local Binary Pattern Histogram* (LBPH) dalam melakukan pengenalan wajah untuk kehadiran secara *real time*. dan melakukan perancangan agar sistem ini dapat berjalan. Pada gambar 3.1 dapat dilihat alur kerja tahapannya.



Gambar 3. 1 Flowchart Proses Sistem Face Recognition

3.1 Implementasi Perangkat

Pada implementasi perangkat akan dijelaskan mengenai perangkat yang akan digunakan dalam membangun sebuah model sistem pengenalan wajah untuk melakukan kehadiran otomatis. Dengan menggunakan metode *Haar Cascade Classifier* dan *Local Binary Patterns Histogram* (LBPH). Perangkat keras (*hardware*) dan perangkat lunak (*software*) yang akan dijelaskan berikut ini.

3.1.1 Perangkat Keras (*Hardware*)

Perangkat keras (*hardware*) yang digunakan untuk membangun sebuah model sistem pengenalan wajah untuk melakukan kehadiran. Secara umum spesifikasi yang perangkat keras yang digunakan pada implementasi tugas akhir ini telah diatas spesifikasi minimum, yang akan ditunjukkan pada tabel 3.1 sebagai berikut.

Tabel 3. 1 Spesifikasi perangkat keras

Perangkat Keras	Spesifikasi yang Digunakan
Processor	Intel® Core™ i7-8565U
Graphic Card	NVIDIA® GeForce® MX150
Hardisk	512GB
RAM	8GB 2400MHz DDR4 Memory
WebCam	720p HD Camera

3.1.2 Perangkat Lunak (*Software*)

Perangkat lunak (*software*) yang digunakan untuk membangun sebuah model diperlukan kesesuaian metode *face recognition* agar dapat berhasil diimplementasi dan di *install* dengan baik serta dapat dipergunakan. Adapun spesifikasi perangkat lunak yang diimplementasikan akan ditunjukkan pada Tabel 3.2 sebagai berikut.

Tabel 3. 2 Spesifikasi perangkat lunak

Perangkat Lunak	Spesifikasi
Operating System	Windows 10 64 bit Home
Microsoft Visual Code	1.63 Version
Python	3.9 Version
Google Chrome	Version 98.0.4758.82 (Official Build) (64-bit)

3.2 Metode Pengumpulan Data

Pada pembuatan sistem ini, tahapan awal yang harus dilakukan adalah membuat *dataset* yang nantinya akan digunakan dijadikan sebagai data yang akan di latih untuk dapat mendeteksi dan mengenali objek. 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 beberapa 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 dan dapat dengan cepat untuk mendeteksi wajah dengan menggunakan metode *Haar Cascade Classifier*.

3.2.1 Pengumpulan Data

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 digunakan untuk *dataset*, 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 atas
- Mengangkat dagu 10° atas

Wajah yang di *capture webcam* tidak terhalangi sebgaiian oleh objek lain dan tidak banyak bergerak agar pengambilan gambar dapat dilakukan dengan baik. Untuk aspek jarak wajah terhadap kamera *webcam* akan dicari jarak ideal yaitu 40 cm sampai dengan 100 cm.

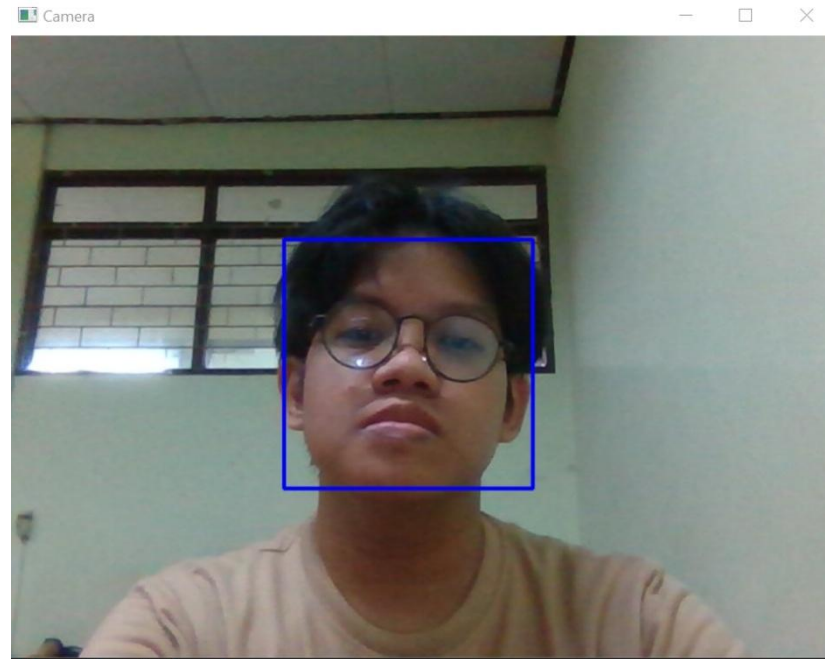
3.3 Pengolahan Citra

Data yang telah dikumpulkan akan melalui beberapa proses sebelum dilanjutkan ke proses training agar dapat melakukan pengenalan. Data yang sudah didapatkan harus dilakukan perubahan citra *image* dan dilakukan *resize* guna untuk meningkatkan performa model saat dilakukan *training* data dan juga untuk dapat mendeteksi objek. Adapun tahap yang akan dilalui adalah *preprocessing*.

3.3.1 *Preprocessing*

Tahap ini dilakukan untuk mempermudah proses *training* data, 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 3.2 sebagai berikut.



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.

Pada perancangan *face detection* atau biasa disebut dengan pendeteksian wajah pada tugas akhir ini menggunakan metode *Haar Cascade Classifier*. Metode tersebut digunakan karena metode ini sangatlah ringan dan juga memiliki kecepatan pendeteksian yang sangat cepat. Maka dengan menggunakan Library yang disediakan oleh OpenCV yaitu *haarcascade frontal_face_default.xml*. yang mana pada *library* tersebut sudah dapat melakukan semua proses pendeteksian wajah.

berikut ini merupakan tahapan cara kerja pada *library* OpenCV *haarcascade frontal_face_default.xml*. yaitu dengan cara berikut ini, Langkah pertama yang harus dilakukan adalah citra harus dikonversikan terlebih dahulu dari RGB menjadi *Grayscale*, Secara teori cara mengkonversikannya adalah dengan cara menggunakan rumus sebagai berikut pada persamaan rumus 3.1

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

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 :

$$\text{Diketahui} = \text{Red (merah)} = 57$$

$$\text{Green(hijau)} = 34$$

$$\text{Blue(biru)} = 16$$

$$\text{Grayscale} = 0.2989 R + 0.5870 G + 0.1140 B$$

$$\text{Grayscale} = (0.2989 \times 57) + (0.5870 \times 34) + (0.1140 \times 16)$$

$$\text{Grayscale} = 17.0373 + 19.968 + 1.824$$

$$\text{Grayscale} = 38.8193$$

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 piksel 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 3.5 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 :

$$NILAI FITUR = |(total piksel hitam) - (total piksel putih)| \quad (3.2)$$

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* :

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

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

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

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

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

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:

<i>Daerah Hitam 1</i> =	2	1	2	3	4	5
	4	5	1	2	3	5
<i>Daerah Putih 1</i> =	3	4	5	5	1	2
	5	1	2	4	3	6
<i>Daerah Hitam 2</i> =	3	7	4	1	8	2
	6	5	1	2	3	4

$$\text{Daerah Hitam 1} = 2 + 1 + 2 + 3 + 4 + 5 + 4 + 5 + 1 + 2 + 3 + 5 = 37$$

$$\text{Daerah Hitam 2} = 3 + 7 + 4 + 1 + 8 + 2 + 6 + 5 + 1 + 2 + 3 + 4 = 46$$

$$\text{Daerah Putih 1} = 3 + 4 + 5 + 5 + 1 + 2 + 5 + 1 + 2 + 4 + 3 + 6 = 41$$

$$\text{NILAI FITUR} = |(\text{total piksel hitam}) - (\text{total piksel putih})|$$

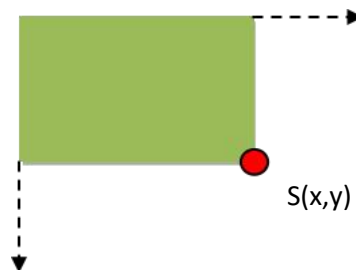
$$\text{NILAI FITUR} = |(37+46) - (41)|$$

$$\text{NILAI FITUR} = 42$$

Cara perhitungan diatas merupakan cara perhitungan manual untuk mencari nilai fitur. Tetapi cara itu tidak akan efektif dan efisien jika ada ratusan jenis fitur *haar* dan juga ratusan jenis ukuran serta posisi fitur yang acak dalam citra. Jika menghitung dengan cara menggunakan *integral image* akan lebih efisien dan dapat dengan sekali jalan. Berikut ini merupakan perhitungan menggunakan *integral image* :

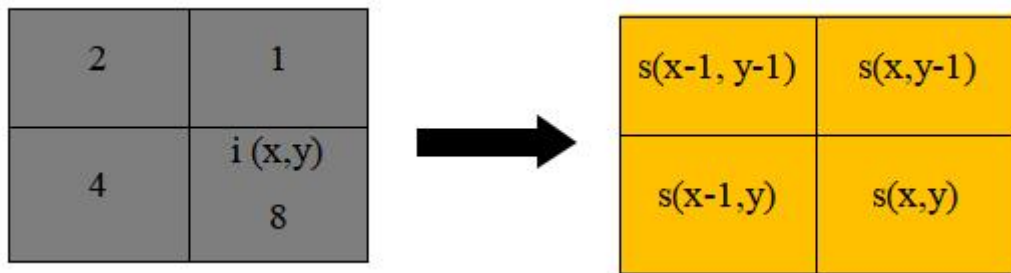
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

1. Menyiapkan matriks dengan ukuran yang sama untuk digunakan sebagai *buffer*. Fungsi dari matriks *buffer* yaitu sebagai tempat untuk meletakkan nilai *integral image* dari setiap piksel.
2. Membuat *summed area table* sebagai contoh seperti pada gambar 3.6 berikut. Fungsi *summed area table* digunakan untuk dapat mempermudah proses perhitungan.



Gambar 3. 6 Ilustrasi area $s(x,y)$

$S(x,y)$ adalah nilai *summed area* pada bidang (x,y) , nilai $i(x,y)$ merupakan intensitas dari citra asli, $s(x,y-1)$ merupakan nilai *summed area* dari nilai piksel tetangga atas atau $(y-1)$. Dan $s(x-1,y)$ merupakan nilai *summed area* dari nilai piksel tetangga x serta $s(x-1,y-1)$ merupakan nilai *summed area* dari nilai piksel tetangga diagonalnya. Jika diilustrasikan dalam gambar akan menjadi seperti gambar 3.7 berikut .



Gambar 3. 7 Summed Area

Gambar 3.7 menunjukkan maka matriks dengan warna *gray* merupakan matriks asli dari citra dan pada warna *orange* merupakan nilai *integral image* yang akan dicari.

3. Selanjutnya adalah mencari nilai *integral image* dengan rumus pada persamaan 3.3. Pada setiap piksel yang berada di citra yang mana ialah setiap nilai pada matriks asli yang telah diketahui dan meletakkan nilainya pada matrik *buffer* yang telah disediakan.

	2	1	2	3	4	5	X
	4	5	1	2	3	5	
	3	4	5	5	1	2	
	5	1	2	4	3	6	
	3	7	4	1	8	2	
	6	5	1	2	3	4	
Y							

$$S(x, y) = i(x, y) + s(x - 1, y) + s(x, y - 1) - s(x - 1, y - 1)$$

$$S(1,1) = 2 + 0 + 0 + 0 + 0 + 0 = 2$$

$$S(x, y) = i(x, y) + s(x - 1, y) + s(x, y - 1) - s(x - 1, y - 1)$$

$$S(2,1) = 2 + 1 + 0 + 0 + 0 + 0 = 3$$

$$S(x, y) = i(x, y) + s(x - 1, y) + s(x, y - 1) - s(x - 1, y - 1)$$

$$S(1, 2) = 2 + 4 + 0 + 0 + 0 + 0 = 6$$

Melakukan perhitungan seperti di atas pada setiap piksel yang ada. Maka hasil setiap perhitungannya dimasukkan ke dalam *buffer* yang telah tersedia sebelumnya. Contoh memasukan *buffer* yang telah dihitung diatas sebagai berikut.

2	3	0	0	0	0
6	12	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Setelah semua terhitung maka akan ditemukan matriks *integral image* yang telah dimasukkan ke dalam *buffer* sebagai berikut.

2	3	5	8	12	17
6	12	15	20	27	37
9	19	27	37	45	57
14	25	35	49	60	78
17	35	49	64	83	103
23	46	61	78	100	124

Setelah didapat nilai dari *integral image*, selanjutnya bagi daerah tersebut menjadi 2 bagian yaitu bagian piksel hitam dan bagian piksel putih sesuai dengan matriks *fitur-haar*.

2	3	5	8	12	17
6	12	15	20	27	37
9	19	27	37	45	57
14	25	35	49	60	78
17	35	49	64	83	103
23	46	61	78	100	124

Untuk menentukan jumlah nilai piksel yang dicari pada sebelumnya dengan menggunakan rumus manual, maka dengan hasil matriks yang telah dihitung menggunakan *integral image* dapat dihitung menggunakan persamaan rumus 3.4 sebagai berikut.

$$\text{Luas Area} = L_1 + L_4 - (L_2 + L_3) \quad (3.4)$$

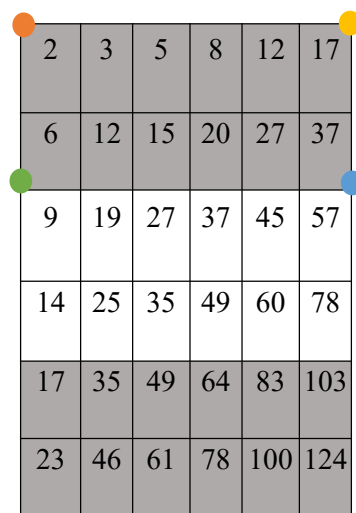
Diketahui = L1 = Orange

L2 = Kuning

L3 = Hijau

L4 = Biru

A. Luas daerah hitam 1



2	3	5	8	12	17
6	12	15	20	27	37
9	19	27	37	45	57
14	25	35	49	60	78
17	35	49	64	83	103
23	46	61	78	100	124

$$\begin{aligned}
 \text{Diketahui } L_1 &= \text{Orange} = 0 \\
 L_2 &= \text{Kuning} = 0 \\
 L_3 &= \text{Hijau} = 0 \\
 L_4 &= \text{Biru} = 37 \\
 \text{Luas area} &= (L_1 + L_4) - (L_2 + L_3) \\
 \text{Luas area} &= 37 - 0 \\
 \text{Luas area} &= 37
 \end{aligned}$$

B. Luas daerah hitam 2

2	3	5	8	12	17
6	12	15	20	27	37
9	19	27	37	45	57
14	25	35	49	60	78
17	35	49	64	83	103
23	46	61	78	100	124

$$\begin{aligned}
 \text{Diketahui } L_1 &= \text{Orange} = 0 \\
 L_2 &= \text{Kuning} = 78 \\
 L_3 &= \text{Hijau} = 0 \\
 L_4 &= \text{Biru} = 124 \\
 \text{Luas area} &= (L_1 + L_4) - (L_2 + L_3) \\
 \text{Luas area} &= 124 - 78 \\
 \text{Luas area} &= 46
 \end{aligned}$$

C. Luas daerah putih

2	3	5	8	12	17
6	12	15	20	27	37
9	19	27	37	45	57
14	25	35	49	60	78
17	35	49	64	83	103
23	46	61	78	100	124

Diketahui = $L_1 = \text{Orange} = 0$

$L_2 = \text{Kuning} = 37$

$L_3 = \text{Hijau} = 0$

$L_4 = \text{Biru} = 78$

$\text{Luas area} = (L_1 + L_4) - (L_2 + L_3)$

$\text{Luas area} = 78 - 37$

$\text{Luas area} = 41$

$NILAI FITUR = |(total \text{ piksel hitam}) - (total \text{ piksel putih})|$

$NILAI FITUR = |(78+37) - (41)|$

$NILAI FITUR = 42$

Setelah didapat nilai dari *integral image* akan terbukti sama dengan perhitungan secara manual sebelumnya. Dimana nilai fitur yang terdapat pada image akan digunakan untuk menentukan objek yang ada dalam kamera, yang kemudian objek yang dideteksi tersebut harus mempunyai nilai *threshold* atau batas ambang nilai fitur.

Setelah didapat sebuah nilai fitur proses terakhir adalah membuat *cascade classifier*. untuk dapat menambah akurasi perlu dilakukan 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 jumlah fiturnya. Jika nilai fitur tidak dapat memenuhi maka hasil langsung ditolak. Contoh *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 *stage* mempunyai *threshold* yang berbeda kemudian kemudian setiap fitur dalam *stage* juga mempunyai *threshold* yang berbeda-beda. Objek yang tidak dapat memenuhi akan ditolak dan yang memenuhi akan melalui *stage* selanjutnya hingga objek ditemukan.

3.4 Pengolahan Data

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 Pattern*. Cara kerjanya adalah dengan mengkonversi seluruh citra *grayscale* yang telah dijadikan *dataset* sebelumnya menjadi *Local Binary Pattern Image* dan dijadikan sebuah *Histogram*. Berikut ini adalah langkah-langkah membuat *Local Binary Pattern*.

1. Langkah pertama menyiapkan citra *grayscale* yang sudah dijadikan *dataset* sebelumnya lalu akan diubah menjadi *local binary pattern image*. Maka Gambar 3.8 merupakan citra *grayscale*.

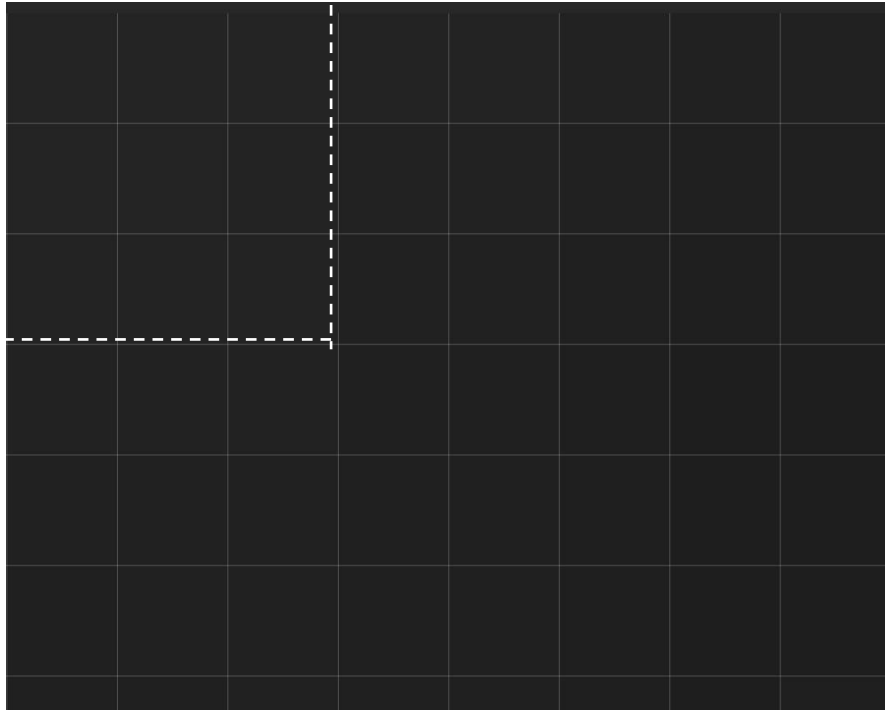


Gambar 3. 8 Citra grayscale

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

$$LBP = \sum_n^c = S(i_n - i_c)2^n \quad (3.3)$$

3. Setelah dilakukan *zoom* hingga terlihat jelas setiap piksel yang diambil matrik 3 x 3 pada piksel, maka gambar akan menjadi seperti pada gambar 3.9



Gambar 3. 9 Matriks 3 x 3 pojok kiri atas

4. Menghitung nilai matrik 3 x 3 yang telah ditandai tersebut dengan menggunakan rumus 3.3. Nilai matriknya adalah sebagai berikut:

1	3	4
6	4	10
3	2	5

Titik pusatnya ialah 4 yang menjadi nilai *threshold* kemudian jika nilai pusat lebih tinggi dari nilai yang ada disekitarnya, maka nilai matriks tersebut dibandingkan dengan matrik disekitarnya, Dengan aturan jika nilai pusat \geq nilai piksel sekitar maka beri nilai 0. Dan jika titik pusat $<$ nilai piksel sekitar maka beri nilai 1. Sesudah dilakukan seperti itu kemudian matriknya akan menjadi seperti berikut ini :

0	0	1
1	Pusat	1
0	0	1

Selesai didapat hasil dari biner tersebut, Maka selanjutnya akan diubah ke dalam bentuk desimal dari hasil biner tersebut. Dengan cara sebagai berikut.

0_7	0_6	1_5
1_0	<i>Pusat</i>	1_4
0_1	0_2	1_3

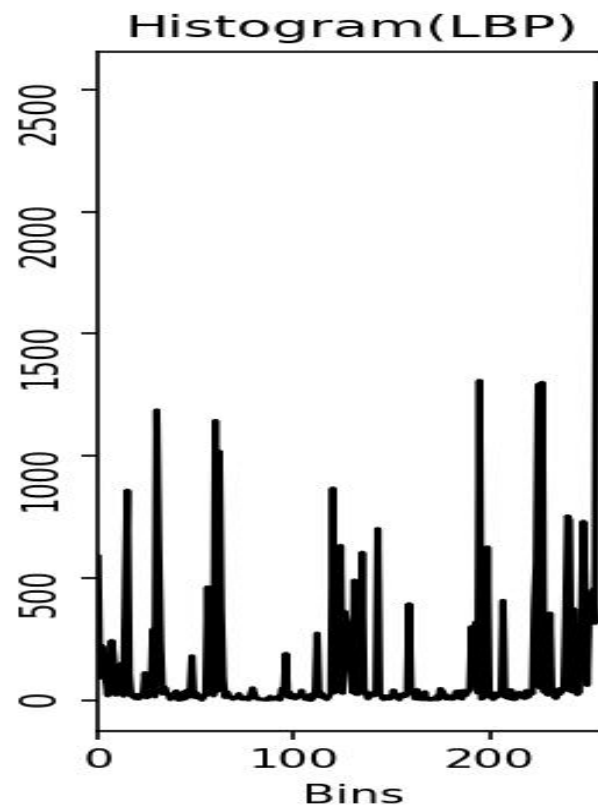
$$\begin{array}{r}
 0_7 \ 0_6 \ 1_5 \ 1_4 \ 1_3 \ 0_2 \ 0_1 \ 1_0 \\
 \hline
 2^5 \ 2^4 \ 2^3 2^0 \\
 \hline
 0 + 0 + 32 + 16 + 8 + 0 + 0 + 1 = 57
 \end{array}$$

Dari hasil perhitungan tersebut diketahui nilai pusatnya adalah 57. Maka perhitungan tersebut akan diimplementasikan pada semua bagian citra maka citra LBP akan menjadi seperti pada gambar 3.10.



Gambar 3. 10 LBP image

Maka hasil perhitungan LBP jika ditampilkan dalam bentuk histogram akan menjadi seperti pada gambar 3.11 sebagai berikut :



Gambar 3. 11 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.

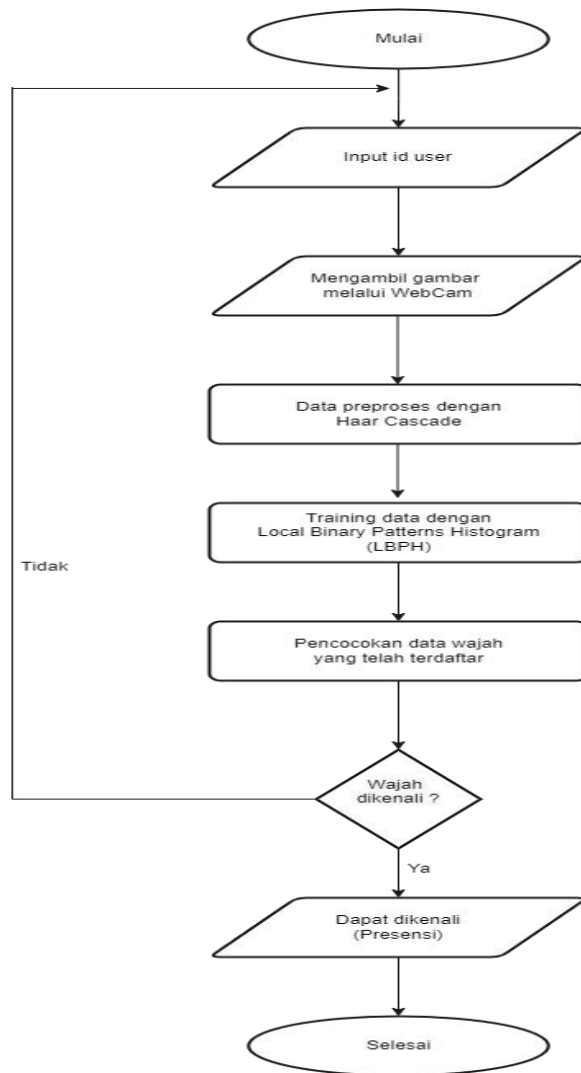
Setelah melalui proses tersebut maka selanjutnya akan membuat algoritma yang dapat menyimpan hasil LBP dalam bentuk yml. Algoritma akan dirancang dalam bentuk *flowchart*. Untuk fungsi *Local Binary Pattern Histogram* menggunakan library OpenCV yaitu `cv2.createRecognizer`. pada *library* tersebut sudah mencakup keseluruhan cara kerja *Local Binary Pattern*.

BAB 4

PEMBAHASAN

4.1 Gambaran Alur Kerja Sistem

Pada pembangunan sistem pengenalan wajah ini menggunakan algoritma *Haar Cascade Classifier* untuk melakukan *preprocessing* agar dapat melakukan pendeteksian wajah yang di olah dengan menggunakan algoritma *Local Binary Pattern Histogram* (LBPH). Berikut gambaran alur kerja sistem pengenalan wajah dapat dilihat pada gambar 4.1 sebagai berikut.



Gambar 4. 1 Flowchart Alur kerja sistem

Berikut uraian setiap proses yang terdapat pada *flowchart*, 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. Menangkap 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 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 dicocokkan menggunakan metode *Haar Cascade Classifier*. Pada tahap ini terdapat perhitungan-perhitungan untuk pengenalan wajah.

6. Wajah dikenali

Wajah dikenali adalah sebuah kondisi apakah wajah 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, tanggal dan waktu saat melakukan presensi.

4.2 Pembuatan Dataset (*Preprocessing*)

Tahap pertama yang dilakukan dalam penelitian ini adalah membangun *Dataset*. *Dataset* yang digunakan dalam implementasi ini berupa *dataset* yang dibuat sendiri dengan mengambil gambar wajah yang ingin diidentifikasi melalui kamera secara *real time*. Setelah itu citra tersebut akan diolah menggunakan metode pendeteksian wajah berupa *Haar Cascade*. Wajah yang terdeteksi ditunjukkan berupa garis bujur sangkar ROI (*Region of Interest*) yang digunakan adalah objek (muka).

Untuk dapat melakukan mengidentifikasi sebuah objek (muka), *user* harus melakukan pengambilan gambar terlebih dahulu secara *real time* sebanyak 100 gambar untuk dijadikan dataset. Maka implementasi *source code* untuk melakukan *open* kamera untuk pengambilan gambar secara *real time*. Akan ditampilkan pada *source code* Gambar 4.2 sebagai berikut.

```
cam = cv2.VideoCapture(1)
cam.set(3, 640) # set video width
cam.set(4, 480) # set video height
```

Gambar 4. 2 Source code untuk menampilkan kamera webcam

Kemudian, akan membaca video *real time* yang ada pada laptop. Setelah itu memberi perintah program agar dapat mendeteksi mana yang merupakan wajah dan mana yang bukan merupakan wajah dengan menggunakan perintah *source code* untuk menjalankan *library* pada gambar 4.3 sebagai berikut:

```
pendeteksi_muka = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Gambar 4. 3 Source code mendeteksi wajah

Kemudian menggambar ROI (*Region Of Interest*) dalam bentuk kotak berwarna biru sesuai dengan koordinat dan dimensinya yaitu (nilai x,y,w,h). maka contoh *source code* pada gambar 4.4 sebagai berikut :

```
cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
```

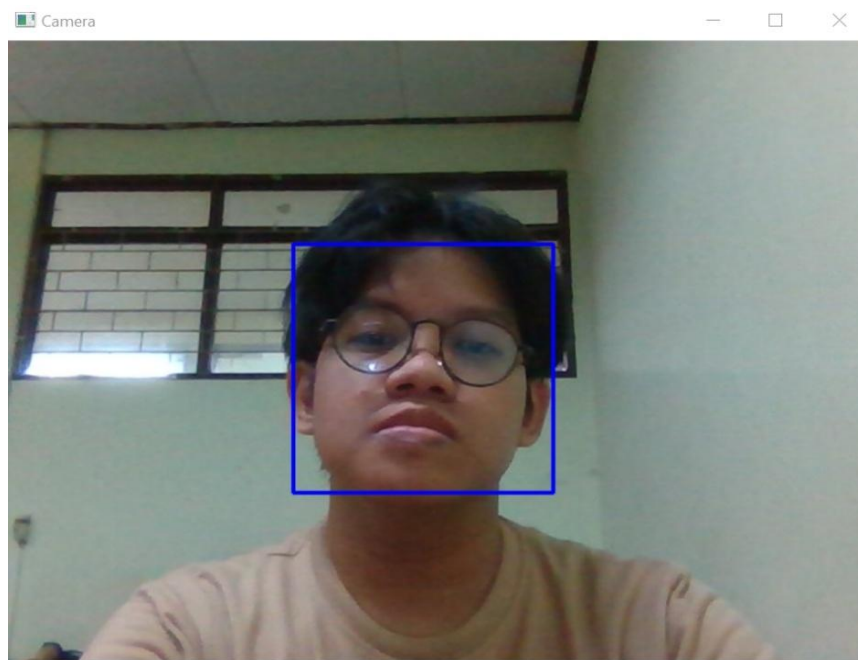
Gambar 4. 4 Source code menggambarkan ROI (*Region Of Interest*)

Kemudian akan meng-*capture* wajah *user* yang akan melakukan identifikasi wajah akan diminta untuk memasukan id untuk dapat menyimpan inisial dari nama *user* yang akan disimpan kedalam dataset, setiap wajah harus ditandai dengan no Id. Maka contoh *source code* pada gambar 4.5 sebagai berikut:

```
face_id = input('\n Masukan nomor id dan tekan enter ==> ')
```

Gambar 4. 5 Source code untuk memasukan Id User

Lalu sistem akan mendeteksi wajah. Jika wajah terdeteksi maka akan ditunjukkan muncul tanda berupa garis bujur sangkar ROI (*Region of Interest*) pada wajah. Lalu akan muncul sebuah tampilan video untuk meng-*capture* wajah *user* sampai 100 gambar. Yang akan ditampilkan pada gambar 4.6 sebagai berikut.



Gambar 4. 6 Pengambilan foto user

Data gambar yang telah melakukan identifikasi wajah akan memproses dan hasil dari pengolahan yang sudah dilakukan tersimpan pada folder “dataset” yang berada dalam *directory* yang sama dengan *source code*. Lalu data akan mentransformasi berbentuk *grayscale*, oleh karena itu masing-masing data gambar yang ditangkap akan dikonversi menjadi *grayscale* akan ditampilkan *source code* pada gambar 4.7 dengan menggunakan fungsi sebagai berikut:









```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Gambar 4. 7 Fungsi mengubah citra RGB menjadi grayscale

4.3 Fitur Pendeksian Wajah

Proses pertama yang dilakukan oleh metode *Haar Cascade Classifier* untuk mendeteksi adanya fitur wajah pada sebuah gambar adalah dengan merubah gambar tersebut menjadi citra *grayscale* dan data yang akan diolah akan dilakukan proses *resize* (mengubah ukuran gambar) diubah menjadi dimensi yang berbeda-beda sesuai dengan rasio wajah yang tertangkap kamera. Proses ini dilakukan untuk dapat memudahkan proses pembelajaran sistem (*training*). Setelah berhasil diambil sebanyak 100 gambar *user* dari hasil yang telah didapat berupa citra *grayscale*, Maka pengolahan yang sudah dilakukan tersimpan pada folder “dataset” yang berada dalam *directory* yang sama dengan *source code*. Maka salah satu contoh hasil dari pengambilan gambar yang telah ditransformasi menjadi citra *grayscale* akan ditampilkan pada Tabel 4.1 sebagai berikut.

Tabel 4. 1 Hasil pengolahan dataset yang sudah menjadi grayscale

Sampel 1		Sampel 2	
			
Sampel 3		Sampel 4	
			
Sampel 5		Sampel 6	



4.4 Proses Pembelajaran Sistem (*Training*)

Sebelum melakukan proses *training*, terdapat modul dari numpy, pillow yang akan digunakan. Modul-modul tersebut akan diimport terlebih dahulu seperti yang ditampilkan pada Gambar 4.8 berikut.

```
import cv2
import numpy as np
from PIL import Image
import os
```

Gambar 4. 8 Importing package dan modul training

Pada proses *training* atau pembelajaran sistem ini akan menggunakan algoritma *Local Binary Patterns Histogram* (LBPH). Algoritma *Local Binary Pattern Histogram* (LBPH) akan dapat mengenali gambar dalam *folder dataset* secara terus-menerus untuk melatih sistem sehingga dapat mengenali gambar wajah tersebut dengan tepat. Maka ditampilkan source code pada Gambar 4.9 sebagai berikut:

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
```

Gambar 4. 9 Metode Local Binary Pattern Histogram

Kemudian membuat suatu fungsi perulangan untuk melatih sistem dalam mengenali gambar secara benar. Berikut adalah perintah *source code* pada gambar 4.10.

```
def dapatGambarDanLabel(path):
```

Gambar 4. 10 Source code melatih dataset

Lalu dibutuhkan perintah untuk mengambil data dari *folder dataset* lalu akan melakukan *training* untuk keseluruhan dan menelusuri gambar yang ada pada *folder dataset*. Yang mana perintah source code pada gambar 4.11 berikut :

```
imagePath = [os.path.join(path,f) for f in os.listdir(path)]
```

Gambar 4. 11 Source code untuk mengambil data yang akan di latih pada dataset

Selanjutnya akan dibuat definisi untuk gambar wajah dan labelnya dengan perintah *source code* pada gambar 4.12 sebagai berikut :

```
sample_muka=[]  
ids = []
```

Gambar 4. 12 Source code untuk membuat definisi wajah dan label

Lalu untuk mempelajari setiap gambar wajah, dapat menggunakan perulangan pada gambar 4.13 perintah *source code* berikut ini , yang mana perintah itu digunakan untuk mengkonversi gambar *grayscale* menjadi bentuk *array*.

```
for imagePath in imagePath:  
PIL_img = Image.open(imagePath).convert('L')
```

Gambar 4. 13 sourcode untuk mengkonversi gambar grayscale menjadi bentuk array



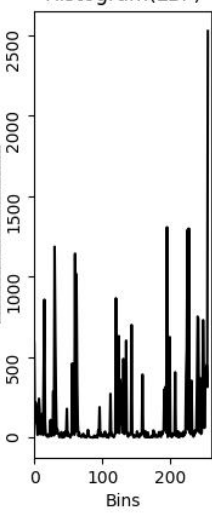


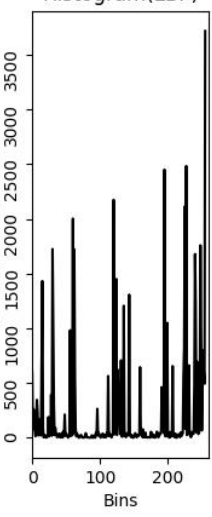
Kemudian dibutuhkan juga modul untuk mengelola yaitu PIL (*Python Imaging Library*) yang akan digunakan untuk membuka, memanipulasi dan menyimpan dari berbagai format *file* gambar dengan perintah *source code* pada gambar 4.14.



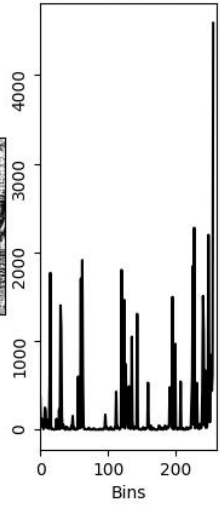


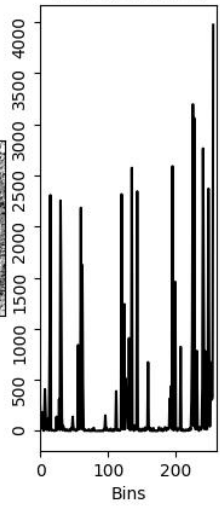
```
PIL_img = Image.open(imagePath).convert('L')  
img_numpy = np.array(PIL_img,'uint8')  
  
for (x,y,w,h) in muka:  
    sample_muka.append(img_numpy[y:y+h,x:x+w])  
    ids.append(id)  
  
return sample_muka,ids
```



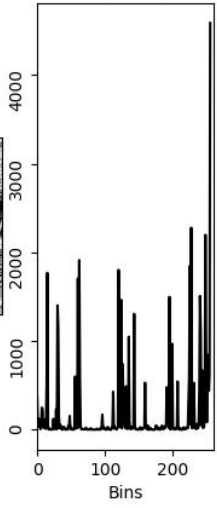


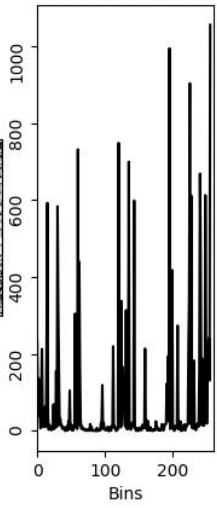
Gambar 4. 14 Souce untuk mengelola Python Imaging Library

Citra wajah pada tabel 4.1 tersebut akan melalui beberapa proses seperti *grayscale* dan juga *local binary pattern histogram*. Berikut beberapa hasil dari proses tersebut dapat dilihat pada tabel 4.2.

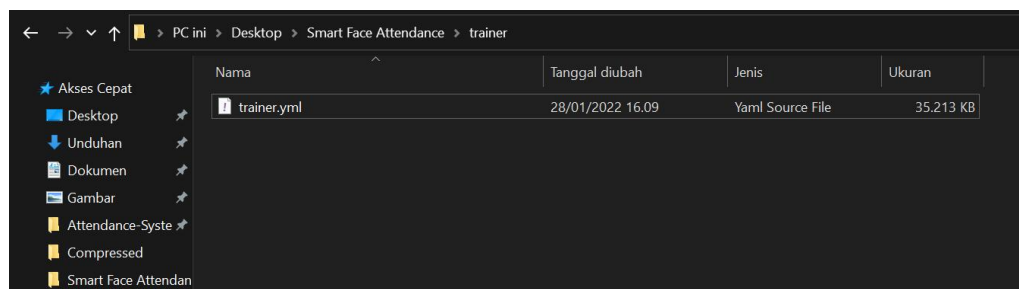
Tabel 4. 2 Hasil pengolahan citra

No	Hasil pengolahan Citra
1.	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>Gray Image</p>  </div> <div style="text-align: center;"> <p>LBP Image</p>  </div> <div style="text-align: center;"> <p>Histogram(LBP)</p>  </div> </div>
2.	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>Gray Image</p>  </div> <div style="text-align: center;"> <p>LBP Image</p>  </div> <div style="text-align: center;"> <p>Histogram(LBP)</p>  </div> </div>

3.	<div data-bbox="539 264 1145 801"><div data-bbox="539 398 715 609"><p>Gray Image</p></div><div data-bbox="750 398 925 609"><p>LBP Image</p></div><div data-bbox="925 264 1145 801"><p>Histogram(LBP)</p></div></div>
4.	<div data-bbox="539 866 1145 1404"><div data-bbox="539 1001 715 1211"><p>Gray Image</p></div><div data-bbox="750 1001 925 1211"><p>LBP Image</p></div><div data-bbox="925 866 1145 1404"><p>Histogram(LBP)</p></div></div>

5.	<div data-bbox="539 264 1145 801"><div data-bbox="539 398 715 609"><p>Gray Image</p></div><div data-bbox="746 398 928 609"><p>LBP Image</p></div><div data-bbox="928 264 1145 801"><p>Histogram(LBP)</p></div></div>
6.	<div data-bbox="539 866 1145 1404"><div data-bbox="539 1001 715 1211"><p>Gray Image</p></div><div data-bbox="746 1001 928 1211"><p>LBP Image</p></div><div data-bbox="928 866 1145 1404"><p>Histogram(LBP)</p></div></div>

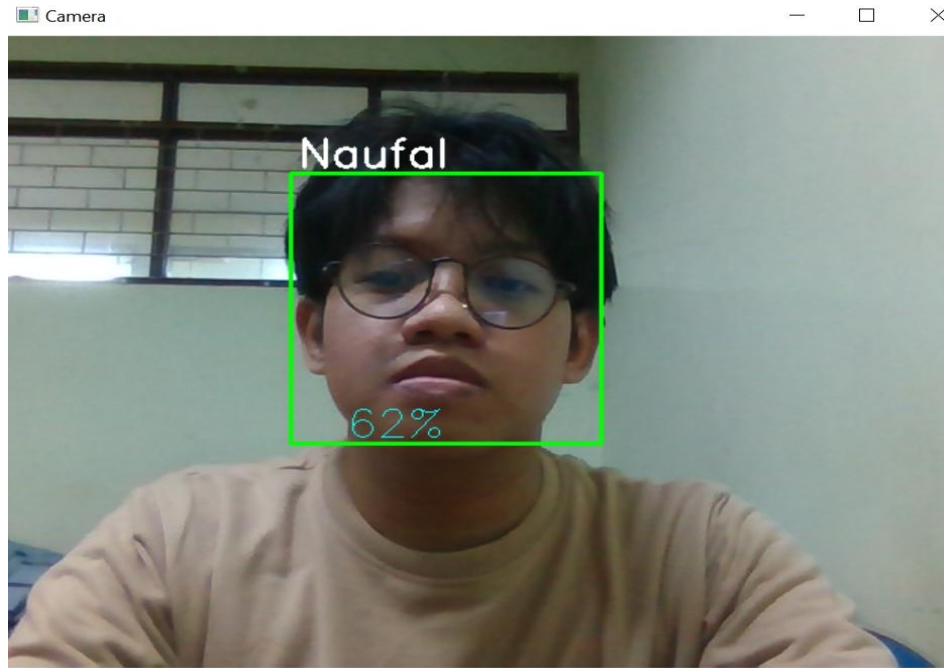
Jika semua pengulangan telah selesai, maka akan disimpan dalam sebuah file dengan ekstensi “trainer.yml”. Selanjutnya dapat mengambil semua data *user* dari *dataset* yang telah dibuat sebelumnya lalu membuat folder “trainer” dalam *directory* yang sama dengan *source code* selanjutnya di proses menggunakan **OpenCV Recognizer** dan dilakukan secara langsung dengan fungsi **OpenCV**. Lalu hasil dari proses training tersebut adalah dalam bentuk file “trainer.yml” yang akan disimpan pada folder “trainer”. Maka untuk hasil training dalam bentuk file “trainer.yml” tersebut akan ditampilkan pada Gambar 4.15



Gambar 4. 15 Directory hasil training dataset

4.5 Fitur Pengenalan Wajah

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 dibandingkan dengan data yang baru. Jika *match* maka wajah akan dapat dikenali sebagai seseorang yang melakukan presensi. Dengan cara menambahkan *variable* nama pada pemilik wajah, akan ditampilkan pada gambar 4.16 berikut ini.



Gambar 4. 16 Face Recognition

4.6 Database Sistem Kehadiran

Dari hasil pendeteksian dan pengenalan yang sudah dilakukan akan mengeluarkan *output* dalam bentuk file .csv yang bernama absen.csv berfungsi sebagai penyimpanan data nama, tanggal dan waktu ketika melakukan presensi. Berikut tampilan *source code* untuk mendata presensi yang masuk pada sistem kehadiran pengenalan wajah ini dapat dilihat pada gambar 4.17

```
def markAttend(name):
    with open('absen.csv', 'r+') as f:
        data = f.readlines()
        name_list = []
        for line in data:
            entry = line.split(',')
            name_list.append(entry[0])
        if name not in name_list:
            date_ = dt.date.today()
            time_ = dt.datetime.today().time().strftime("%H:%M:%S")
            f.writelines(f'\n{name},{time_},{str(date_.strftime("%A-%B-%d-%Y"))}')
```

Gambar 4. 17 Source code untuk mendata presensi

Maka akan mengeluarkan *output* sebuah file absen.csv yang berisi daftar user yang telah berhasil hadir melalui sistem kehadiran pengenalan wajah ini. Dapat dilihat pada gambar 4.18

1	Name , Date, Time								
2									
3	Naufal,15:23:39,Friday-August-12-2022								
4	Bayu,16:37:31,Friday-August-12-2022								
5	Alif,16:41:58,Friday-August-12-2022								
6	Rasya,16:25:40,Sunday-August-14-2022								
7	Nurahmawati,16:38:42,Sunday-August-14-2022								
8	Ivan,11:00:38,Monday-August-15-2022								
9	Rafli,11:01:16,Monday-August-15-2022								
10	Kevin,14:10:43,Monday-August-15-2022								
11									
12									
13									
14									
15									

Gambar 4. 18 Tampilan data yang berhasil melakukan presensi

4.7 Pengujian Sistem Identifikasi Wajah

Dalam proses pengujian (*testing*) ini dilakukan saat menguji akurasi dan kesesuaian metode *face recognition* untuk dapat mengidentifikasi seseorang. Tujuan dari pengujian dari proses ini adalah untuk memvalidasi bahwa sistem yang dibangun berjalan dengan baik.

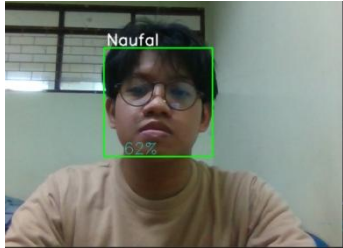
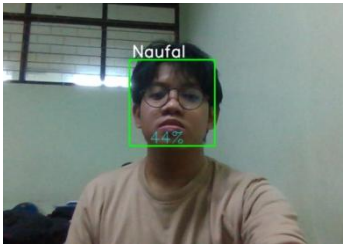


Proses ini dilakukan dengan metode *Haar cascade classifier* dan *Local Binary Pattern Histogram* (LBPH) dengan menggunakan 6 Sampel wajah yang berbeda. Proses pengujian ini akan dilakukan dengan cara merekam atau meng-*capture* wajah *user* sebanyak 100 gambar lalu merubah gambar menjadi *grayscale*. Kemudian akan tersimpan kedalam folder “dataset” yang berada dalam *directory* yang sama dengan *source code* selanjutnya kumpulan *dataset* tersebut akan dilatih menggunakan metode *Local Binary Pattern Histogram* (LBPH), setelah berhasil dilatih akan tersimpan pada folder trainer dan hasil pelatihan dataset tadi akan mengeluarkan *output* berupa file “trainer.yml”. Lalu selanjutnya menjalankan file “face_recognition” untuk melakukan pengujian identifikasi wajah dapat dilakukan ketika proses training dataset telah berhasil.

Akan dilakukan pengujian akurasi pengenalan wajah akan disajikan dalam bentuk tabel agar lebih mudah dimengerti oleh pembaca. Tabel penyajian akan memiliki 4 kolom yaitu no, jarak, hasil, dapat mengenali wajah ya/tidak. Untuk dapat melakukan pengujian menggunakan parameter seperti berikut ini:





- A. Pengaruh jarak terhadap tingkat keberhasilan dan akurasi pengenalan wajah yang sudah tersimpan pada *database*. Tabel 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 merupakan hasil uji coba terhadap jarak untuk pengenalan wajah.
- B. Pengaruh tingkat kemiringan wajah terhadap keberhasilan dan akurasi pengenalan wajah yang sudah tersimpan di *database*. Tabel 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 merupakan hasil dari pengujian terhadap kemiringan wajah untuk pengenalan wajah.

A. Parameter Pertama

Tabel 4. 3 Hasil pengujian sampel

No	Jarak	Hasil	Mengenal Wajah	
			Ya	Tidak
1.1	40 cm		✓	
1.2	70 cm		✓	
1.3	100 cm		✓	
1.4	180 cm			✓

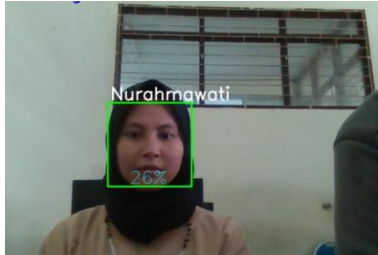

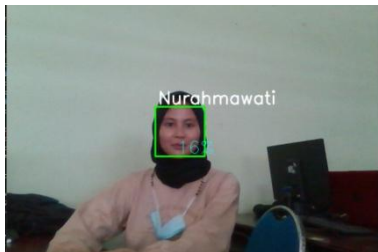

Tabel 4. 4 Hasil pengujian sampel 2

No	Jarak	Hasil	Mengenal Wajah	
			Ya	Tidak
1.2.1	40 cm		✓	
1.2.2	70 cm		✓	
1.2.3	100 cm		✓	
1.2.4	180 cm			✓

Tabel 4. 5 Hasil Pengujian Sampel 3

No	Jarak	Hasil	Mengenal Wajah	
			Ya	Tidak
1.3.1	40 cm		✓	
1.3.2	70 cm		✓	
1.3.3	100 cm		✓	
1.3.4	180 cm			✓

Tabel 4. 6 Hasil Pengujian Sampel 4

No	Jarak	Hasil	Mengenal Wajah	
			Ya	Tidak
1.4.1	40 cm		✓	
1.4.2	70 cm		✓	
1.4.3	100 cm		✓	
1.4.4	180 cm			✓

Tabel 4. 7 Hasil Pengujian Sampel 5


No	Jarak	Hasil	Mengenai Wajah	
			Ya	Tidak
1.5.1	40 cm		✓	
1.5.2	70 cm		✓	
1.5.3	100 cm		✓	
1.5.4	180 cm			✓

Tabel 4. 8 Hasil pengujian sampel 6



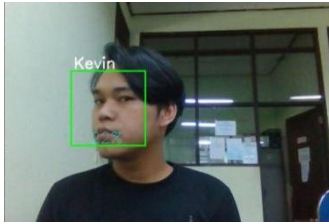
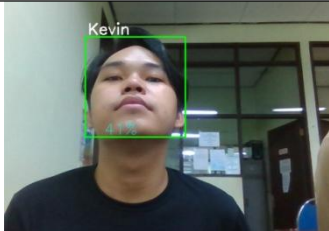

No	Jarak	Hasil	Mengenai Wajah	
			Ya	Tidak
1.6.1	40 cm		✓	
1.6.2	70 cm		✓	
1.6.3	100 cm		✓	
1.6.4	180 cm			✓

B. Parameter Kedua





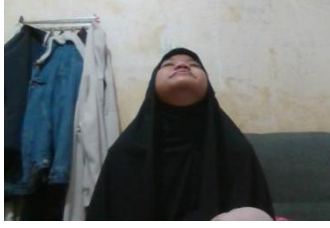
Tabel 4. 9 Hasil Pengujian Sampel 1, Parameter ke 2

No	Drajat kemiringan wajah	Hasil	Mengenal Wajah	
			Ya	Tidak
2.1.1	Tegak lurus		✓	
2.1.2	10° Ke kanan		✓	
2.1.3	10° Ke kiri		✓	
2.1.4	10° Ke atas		✓	
2.1.5	40° Ke atas			✓

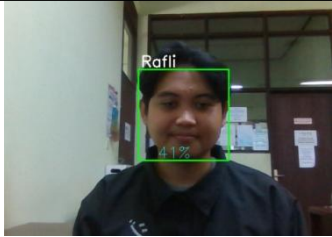

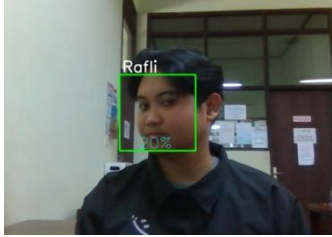


Tabel 4. 10 Hasil Pengujian Sampel 2, Parameter ke 2

No	Drajat kemiringan wajah	Hasil	Mengenai Wajah	
			Ya	Tidak
2.2.1	Tegak lurus		✓	
2.2.2	10° Ke kanan		✓	
2.2.3	10° Ke kiri		✓	
2.2.4	10° Ke atas		✓	
2.2.5	40° Ke atas			✓

Tabel 4. 11 Hasil Pengujian Sampel 3, Parameter ke 2

No	Drajat kemiringan wajah	Hasil	Mengenal Wajah	
			Ya	Tidak
2.3.1	Tegak lurus		✓	
2.3.2	10° Ke kanan		✓	
2.3.3	10° Ke kiri		✓	
2.3.4	10° Ke atas		✓	
2.3.5	40° Ke atas			✓

Tabel 4. 12 Hasil Pengujian Sampel 4, Parameter ke 2

No	Drajat kemiringan wajah	Hasil	Mengenai Wajah	
			Ya	Tidak
2.4.1	Tegak lurus		✓	
2.4.2	10° Ke kanan		✓	
2.4.3	10° Ke kiri		✓	
2.4.4	10° Ke atas		✓	
2.4.5	40° Ke atas			✓

Tabel 4. 13 Hasil Pengujian Sampel 5, Parameter ke 2

No	Drajat kemiringan wajah	Hasil	Mengenal Wajah	
			Ya	Tidak
2.5.1	Tegak lurus		✓	
2.5.2	10° Ke kanan		✓	
2.5.3	10° Ke kiri		✓	
2.5.4	10° Ke atas		✓	
2.5.5	40° Ke atas			✓

Tabel 4. 14 Hasil Pengujian Sampel 6, Parameter ke 2

No	Drajat kemiringan wajah	Hasil	Mengenali Wajah	
			Ya	Tidak
2.6.1	Tegak lurus		✓	
2.6.2	10° Ke kanan		✓	
2.6.3	10° Ke kiri		✓	
2.6.4	10° Ke atas		✓	
2.6.5	40° Ke atas			✓

4.7.1 Hasil Pengujian Analisa

A. Parameter Pertama

Pada parameter pertama menunjukkan bahwa terdapat wajah yang dapat dideteksi dan dikenali pada jarak 40 cm hingga pada jarak 200 cm sudah tidak dapat mengenali wajah tetapi masih dapat mendeteksi wajah.

Tabel 4. 15 Tabel data pengujian

Total Pengujian Benar	20
Total Pengujian Salah	3
Total Data Pengujian	24

Lalu berdasarkan Tabel 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 yaitu dapat disimpulkan bahwa dari 6 sampel data wajah orang yang berbeda, terdapat 20 pengujian yang dapat dikenali oleh sistem dengan benar, dan terdapat 4 pengujian gambar yang tidak dapat dikenali tapi masih bisa mendeteksi wajah.

Cara untuk menghitung nilai akurasi dan kesalahan dari proses pengujian diatas dapat dihitung dengan menggunakan persamaan berikut ini:

$$\text{Akurasi} = \frac{\text{Jumlah data yang benar}}{\text{Jumlah seluruh data}} \times 100\%$$

$$\text{Akurasi} = \frac{20}{24} \times 100\%$$

$$\text{Akurasi} = 83.3\%$$

Sedangkan proses perhitungan untuk data yang tidak berhasil dikenali atau dideteksi adalah sebagai berikut.

$$\text{Kesalahan} = \frac{\text{Jumlah data yang salah}}{\text{Jumlah seluruh data}} \times 100\%$$

$$\text{Kesalahan} = \frac{4}{24} \times 100\%$$

$$\text{Akurasi} = 16.67\% \text{ atau } 17\%$$

Dari perhitungan akurasi dan kesalahan di atas, didapat bahwa persentase nilai akurasi sistem pengenalan wajah untuk melakukan kehadiran otomatis sebesar 83.33% dengan kesalahan sebesar 16.67% atau jika dibulatkan menjadi 17%.

B. Parameter Kedua

Pada parameter kedua menunjukkan bahwa tingkat kemiringan mempengaruhi pengenalan wajah. Dalam uji coba ini akan melakukan kemiringan wajah ke kanan, ke kiri dan ke atas. Lalu ketika tingkat kemiringan kurang lebih 40 derajat wajah sudah tidak bisa dideteksi dan dikenali.

Tabel 4. 16 Tabel data pengujian parameter ke 2

Total Pengujian Benar	24
Total Pengujian Salah	6
Total Data Pengujian	30

Lalu berdasarkan dari tabel 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 yaitu dapat disimpulkan bahwa dari 6 sampel data wajah orang yang berbeda terdapat 30 pengujian yang dapat dikenali oleh sistem dengan benar, dan terdapat 6 pengujian gambar yang tidak dapat terdeteksi.

Cara untuk menghitung nilai akurasi dan kesalahan dari proses pengujian diatas dapat dihitung dengan menggunakan persamaan berikut ini.

$$\text{Akurasi} = \frac{\text{Jumlah data yang benar}}{\text{Jumlah seluruh data}} \times 100\%$$

$$\text{Akurasi} = \frac{24}{30} \times 100\%$$

$$\text{Akurasi} = 80.00\%$$

Sedangkan proses perhitungan untuk data yang tidak berhasil dikenali atau dideteksi adalah sebagai berikut.

$$Kesalahan = \frac{Jumlah\ data\ yang\ salah}{Jumlah\ seluruh\ data} \times 100\%$$

$$Akurasi = \frac{6}{30} \times 100\%$$

$$Akurasi = 20.00\%$$

Dari perhitungan akurasi dan kesalahan di atas, didapat bahwa persentase nilai akurasi sistem pengenalan wajah untuk melakukan kehadiran otomatis sebesar 80.00% dengan kesalahan sebesar 20.00% atau jika dibulatkan menjadi 20%.

BAB 5

PENUTUP

5.1 Kesimpulan

Terdapat beberapa kesimpulan yang didapatkan dari uji coba implementasi sistem pendeteksi pengenalan wajah dengan algoritma *Local Binary Pattern Histogram* (LBPH) adalah sebagai berikut:

1. Algoritma *Local Binary Pattern Histogram* (LBPH) dapat diimplementasikan untuk membangun sebuah model sistem pendeteksian pengenalan wajah untuk melakukan presensi sehingga mampu melakukan pengenalan wajah dengan tingkat akurasi dari hasil uji coba yaitu sebesar **83.3 %** dengan persentase kesalahan sebesar **16.67%** dalam uji coba jarak yang dapat dilakukan untuk mengenalkan wajah seseorang, sedangkan untuk melakukan pengenalan wajah berdasarkan tingkat kemiringan wajah, mempunyai hasil uji coba yaitu sebesar **80.00%** dengan persentase kesalahan sebesar **20.00%**.
2. Tingkat akurasi dapat dipengaruhi dengan adanya banyak sampel data yang akan di-*training*, yang mana semakin banyak data yang dimiliki maka tingkat akurasi akan semakin tinggi.

5.2 Saran

Penelitian tugas akhir ini masih belum sempurna, terdapat beberapa hal yang dapat diperbaiki dan dikembangkan dari penelitian ini. Adapun saran untuk pengembangan penelitian sistem pendeteksian pengenalan wajah berikutnya adalah sebagai berikut:

1. Menggunakan metode *deep learning* lainnya dalam mengimplementasi sistem pengenalan wajah untuk melakukan kehadiran otomatis untuk dibandingkan tingkat keakuratannya.
2. Model yang sudah dibangun dapat dikembangkan dalam aplikasi *mobile* maupun *website*.