SKRIPSI

SISTEM PRESENSI PEGAWAI BERBASIS COMPUTER VISION

Disusun dan diajukan oleh:

MUHAMMAD KAHFI FAJRI KUDDUS

D42114313

DEPARTEMEN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

MAKASSAR

2021

LEMBAR PENGESAHAN SKRIPSI

SISTEM PRESENSI PEGAWAI BERBASIS COMPUTER VISION

Disusun dan diajukan oleh

MUHAMMAD KAHFI FAJRI KUDDUS

D421 14 313

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi

Program Sarjana Program Studi Informatika

Fakultas Teknik Universitas Hasanuddin

Pada tanggal 14 Oktober 2021

dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui

Pembimbing Utama,	Pembimbing Pendamping,
Dr. Indrabayu, S.T.,M.T., M.Bus, Sys	A. Ais Prayogi Alimuddin, ST., M.Eng
Nip. 19750716 200212 1 004	Nip. 19830510 201404 1 001

Ketua Program Studi

Dr. Amil Ahmad Ilham, S.T., M.IT

Nip. 19731010 199802 1 001

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini :

Nama : MUHAMMAD KAHFI FAJRI KUDDUS

Nim : D421 14 313

Program Studi : S1 Teknik Informatika

Menyatakan dengan sebenar-benarnya bahwa skripsi yang berjudul :

SISTEM PRESENSI PEGAWAI BERBASIS COMPUTER VISION

Adalah karya ilmiah saya sendiri dan sepanjang pengetahuan saya di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan/ditulis/diterbitkan sebelumnya, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila dikemudian hari ternyata didalam naskah skripsi ini terdapat unsur-unsur djiplakan, saya bersedia menerima sanksi atas perbuatan tersebut dan diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2000, pasal 25 ayat 2 dan pasal 70).

Makassar, 14 Oktober 2021

Yang membuat Pernyataan

MUHAMMAD KAHFI FAJRI KUDDUS

KATA PENGANTAR

Assalamu Alaikum Wr. WB.

Segala puji dan syukur atas kehadirat Allah SWT yang telah melimpahkan Rahmat dan Karunia-Nya sehingga Tuhas Akhir dengan judul **"SISTEM PRESENSI PEGAWAI BERBASIS COMPUTER VISION**" dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jejang Strata-1 pada Departemen Teknik Informatika Universitas Hasanuddin.

Pada penyusunan kali ini disajikan hasil penelitian menyagkut judl yang telah diangat dan telah melalui proses pencarian dari berbagai sumber baik jurnal penelitian, buku maupun dari situs-situs internet, dan dari hasil-hasil penelitian sebelumnya.

Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai dengan masa penyusunan Tugas Akhir, sangatlah sulit untuk menyelesaikan Tugas Akhir ini. Oleh karena itu, penulis berterima kasih kepada:

1) Kedua Orang tua penulis, Bapak Ir. Muhammad Kuddus MM. dan Ibu Drs. Rakidah Arifuddin serta saudara-saudara dan keluarga penulis, yang selau memberikan dukungan, doa dan semangat;

2) Bapak Dr. Indrabayu ST., MT., M.Bus.Sys., selaku pembimbing 1 dan bapak A. Ais Prayogi Alimuddin, S.T., M.Eng., selaku pembimbing II yang selalu menyediakan waktu, tenaga, pikiran dan perhatian yang luar biasa untuk mengarahkan penulis dalam penyusunan Tugas Akhir;

3) Bapak Amil Ahmad Ilham, ST., M.IT., Ph.D selaku Ketua Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin atas bimbingannya selama masa perkuliahan penulis;

4) Bapak H. Saldy Mansyur, SE.Ak., MM., CWM selaku paman saya yang selalu memberikan saya semangat untuk menyelesaikan tugas akhir ini sampai tuntas;

5) Para Sahabat, teman-teman tim *face recognition*, teman-teman dan kakak-kakak AIMP *Research Group* Unhas terutama kepada Sofyan Tandungan, Tiwi Nur Safitri, Angela Hervina Gosal, Syahdan Edy Murad, Magfira Putri Rachmat, Rahmiyanti Rusli, Rizka Irianty, Erlangga, Anisah Mayang Sari, Fadel Pratama dan Muh. Arkan Musyabbir yang telah memberikan begitu banyak bantuan, semangat, inspirasi, ilmu dan hiburan selama penelitian, pengambilan data dan diskusi progress penyusunan Tugas Akhir;

6) Segenap Staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah membantu penulis;

6) Seluruh teman-teman angkatan 2014 Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin;

8) Orang-orang berpengaruh lainnya yang tanpa sadar telah menjadi inspirasi penulis.

Akhir kata, penulis berharap semoga Allah SWT. berkenan membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga Tugas Akhir ini dapat memberikan manfaat bagi pengembangan ilmu. Aamiin.

Wassalam
Makassar, Oktober 2021

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Penelitian tentang presensi pegawai ada beberapa macam cabang yang meliputi semua karakteristik biometrik manusia seperti wajah, iris, sidik jari, pembuluh darah jari, bibir, suara, dan masing-masing dari karakteristik memiliki kelebihan dan kekurangan masing-masing. Khususnya untuk wajah dimana wilayah biometrik ini dapat melakukan identifikasi massal yang biasanya tidak dapat dilakukan oleh karakteristik biometrik lain.. Attendance Management System (AMS) automatis ini berdasarkan teknik pendeteksian dan pengenalan wajah. Kamera dengan resolusi tinggi diletakkan di atas papan tulis menghadap siswa untuk mengambil gambar, Gambar yang diterima akan ditransmisi ke server dan akan disimpan dan diproses secara real time Dari penelitian di atas, pendeteksian wajah di lakukan dengan kondisi ruangan tertentu dan juga karyawan ditetapkan harus berada di dalam kamera untuk menandakan dia hadir dan juga sistem ini hanya mendeteksi kehadiran. Oleh karena itu pada penelitian ini akan dibuat sebuah sistem yang hanya mendeteksi wajah karyawan pada saat kedatangan dan kepulangan. Metode yang diterapkan dalam penelitian ini adalah studi literature terkait, identifikasi kebutuhan penelitian, pengambilan data citra/wajah pegawai, praproses data, implementasi algoritma visi computer dalam data video, pengujian ketepatan deteksi wajah dan rekognisi wajah , analisis akurasi rekognisi wajah. Hasil dari penelitian menyeluruh ini dapat dikatakan bahwa sistem berjalan dengan yang di inginkan dan sesuai dengan yang dirancangkan. Tingkat akurasi dari pengenalan wajah dalam system ini sebesar 96.36%

Kata kunci: Attendance Management System , Biometrik, Kamera, Wajah.

ABSTRACT

Research on employee absenteeism has several branches covering all human biometric characteristics such as face, iris, fingerprint, finger blood vessels, lips, voice, and each of these characteristics has its own advantages and disadvantages. Especially for faces where this biometric area can perform mass identification which usually cannot be done by other biometric characteristics. This automatic Attendance Management System (AMS) is based on face detection and recognition techniques. A high-resolution camera is placed on the blackboard facing students to take pictures, the received image will be transmitted to the server and will be stored and processed in real time. From the research above, face detection is carried out with certain room conditions and employees are set to be in in the camera to indicate he is present and also this system only detects the presence. Therefore, in this study, a system will be created that only detects the faces of employees upon arrival and departure.The method applied in this research is the study of related literature, identification of research needs, retrieval of employee image/face data, data preprocessing, implementation of computer vision algorithms in video data, testing the accuracy of face detection and facial recognition, analysis of facial recognition accuracy. The results of this thorough research can be said that the system runs as desired and in accordance with what was designed. The accuracy rate of facial recognition in this system is 96.36%

Keywords: Attendance Management System, Biometrics, Camera, Face.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi komputer sekarang sangat pesat, ini ditandai dengan hampir semua pengolahan data dan informasi telah dilakukan dengan komputer. Hal ini diakibatkan semakin beraneka ragam permasalahan informasi yang harus ditangani, salah satunya adalah dalam hal presensi pegawai.

Perusahaan ritel XYZ telah menggunakan salah satu sistem presensi yang mengimplementasikan teknologi *Radio-Frequency Identification* (RFID). Tapi karena banyaknya karyawan dan keterbatasan alat yang digunakan sehingga menyebabkan antrian yang panjang ketika melakukan presensi rutin karena setiap karyawan harus melakukan prosedur presensi dengan alat yang terbatas dan prosedurnya memakan waktu. Oleh karena itu dibutuhkannya sistem presensi yang lebih efektif.

Penelitian tentang presensi pegawai ada beberapa macam cabang yang meliputi semua karakteristik biometrik manusia seperti wajah, iris, sidik jari, pembuluh darah jari, bibir, suara, dan masing-masing dari karakteristik memiliki kelebihan dan kekurangan masing-masing. Khususnya untuk wajah dimana wilayah biometrik ini dapat melakukan identifikasi massal yang biasanya tidak dapat dilakukan oleh karakteristik biometrik lain. Dan juga sistem biometrik ini tidak memerlukan kontak langsung pada orang yang diverifikasi identitasnya. Karena inilah sistem ini sangat cocok untuk pemantauan, pelacakan, dan di sistem

otomatisasi [1]. Ini menjadi alasan sistem biometrik wajah sangat cocok digunakan di perusahaan ritel XYZ.

Pada penelitian sebelumnya memberikan solusi dengan membuat perancangan sistem presensi siswa automatis untuk menghilangkan adanya proses kerja manual yang memakan waktu. *Attendance Management System* (AMS) automatis ini berdasarkan teknik pendeteksian dan pengenalan wajah. Kamera dengan resolusi tinggi diletakkan di atas papan tulis menghadap siswa untuk mengambil gambar, Gambar yang diterima akan ditransmisi ke *server* dan akan disimpan dan diproses secara *real time* [2].

Dari penelitian di atas, pendeteksian wajah di lakukan dengan kondisi ruangan tertentu dan juga karyawan ditetapkan harus berada di dalam kamera untuk menandakan dia hadir dan juga sistem ini hanya mendeteksi kehadiran. Oleh karena itu pada penelitian ini akan dibuat sebuah sistem yang hanya mendeteksi wajah karyawan pada saat kedatangan dan kepulangan.

Dengan demikian, dalam penelitian ini akan diangkat judul "Sistem Presensi pegawai berbasis *computer vision*".

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka pokok permasalahan dalam penelitian ini yaitu:

a. Bagaimana cara mengenali wajah karyawan menggunakan *computer vision*?

b. Bagaimana cara mendata karyawan yang telah hadir dan yang telah istirahat menggunakan *computer vision*?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang diuraikan, maka tujuan dari penelitian ini yaitu :

1. Mengetahui cara mengenali wajah karyawan menggunaan *computer vision.*
2. Mengetahui cara mendata kehadiran karyawan yang telah hadir dan yang telah beristirahat menggunakan *computer vision.*

1.4 Batasan Masalah

Ruang lingkup pembahasan tugas akhir ini dibatasi hanya mencakup hal-hal berikut:

a. Objek penelitian adalah perusahaan retail XYZ
b. Wajah yang di deteksi merupakan wajah yang menghadap kearah kamera.
c. Wajah yang di deteksi memiliki penerangan yang baik
d. Pengambilan data pegawai menggunakan Kamera Penguji.
e. Jenis kamera yang digunakan adalah :
    - Webcam Logitech C922 Pro

1.5 Sistematika Penulisan

Adapun sistematika penulisan penelitian adalah sebagai berikut :

BAB I : PENDAHULUAN

Bab ini menguraikan secara singkat latar belakang penelitian, rumusan dan batasan masalah, tujuan dan manfaat penelitian, serta sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Bab ini membahas tentang kerangka berpikir, serta landasan teori yang berhubungan proses perancangan sistem presensi pegawai menggunakan metode *computer vision*.

III : METODOLOGI PENELITIAN

Bab ini membahas tentang jenis penelitian, metode pengumpulan data, alat dan bahan penelitian, metode pengujian dan hasil penelitian.

BAB IV : HASIL DAN PEMBAHASAN

Bab ini membahas tentang hasil data yang dihasilkan dari perancangan sistem presensi pegawai menggunakan metode *computer vision*.

BAB V : PENUTUP

Bab ini merupakan bab penutup yang berisi kesimpulan dan saran-saran.

1.6 Manfaat Penelitian

Diharapkan dengan melakukan penelitian ini dapat diambil manfaat sebagai berikut:

1. Bagi perusahaan, dapat memiliki sistem presensi karyawan secara automatis yang dapat mengurangi waktu tunggu karyawan untuk melakukan presensi.

2. Bagi peneliti, dapat digunakan untuk menanmbah pengetahuan dan sebagai referensi mengenai sistem presensi menggunakan metode pengolahan citra.
3. Bagi institusi pendidikan, dapat digunakan sebagai referensi dalam pengembangan penelitian topic terkait untuk sistem presensi menggunakan metode pengolahan citra.

BAB II

TINJAUAN PUSTAKA

2.1 Pengertian Wajah

Menurut KBBI (Kamus Besar Bahasa Indonesia), wajah merupakan bagian depan dari kepala. Wajah merupakan bagian dari tubuh manusia yang penting pada saat berinteraksi langsung kepada sesame manusia lainnya. Wajah memainkan peranan vital dengan menunjukkan identitas sesorang. Wajah terutama digunakan untuk ekspresi wajah, penampilan, serta identitas. Tidak ada satu wajahpun yang serupa mutlak dengan lainnya, sehingga wajah dapat menjadi salah satu tanda pengenal ciri khusus pada seseorang. Wajah atau muka adalah bagian depan dari kepala pada manusia meliputi wilayah dari dahi (bagian atas) hingga dagu (bagian bawah). Bagian-bagian yang termasuk wilayah wajah yaitu rambut, dahi, alis, mata, pipi, hidung, kumis, mulut, bibir, gigi, kulit, janggut dan dagu.

Dalam ilmu kesehatan wajah dijelaskan berdasakan karakteristik anatominya merupakan bagian *anterior* (struktur bagian depan) dari kepala, dengan batas kedua telinga di *lateral* (struktur terjauh dari garis pertengahan tubuh), dagu di *inferior* (bagian terendah) dan garis batas tumbuhnya rambut di *superior* (bagian teratas). Wajah merupakan bagian dari tubuh manusia yang bisa dikenali dengan teknologi biometrik untuk mengenal suatu individu tertentu. Pada teknologi biometric ciri-ciri biologis dari manusia seperti wajah bisa memberikan informasi yang unik. Pada wajah sendiri terdapat 66 fitur titik yang dapat menjadi ciri khusus pada suatu wajah (Kalansuriya & Dharmaratne, 2013) [3]. Informasi

unik tersebut dapat berupa karakteristik dari pola wajah sesuai dengan individu masing-masing, karena karakteristik dari pola wajah bisa diukur dan dianalisis untuk proses deteksi maupun autentifikasi. Sehingga wajah dapat digunakan sebagai salah satu ciri khusus untuk mengenal seorang individu. Fitur-fitur atau bagian-bagian pada wajah digunakan untuk menjadi ciri khusus pembeda laki-laki dan perempuan (bukan laki-laki).

Perbandingan pada mata dan mulut. Mata laki-laki memiliki ukuran mata yang kecil dan jarak mata dengan alis lebih dekat, sedangkan mata perempuan (bukan laki-laki) memiliki ukuran mata yang lebih besar dan jarak mata dengan alis lebih besar. Sedangkan pada fitur mulut atau bibir, mulut laki-laki memiliki jarak dasar hidung ke bibir atas lebih jauh dan ukuran yang lebih tipis. Sedangkan pada mulut atau bibir perempuan (bukan laki-laki) jarak dasar hidung ke bibir atas lebih sedikit dan ukurannya lebih tebal (Wulansari, Djamal, & Ilyas, 2017) [4]. Bentuk hidung laki-laki umumnya lebih besar sedangkan bentuk hidung perempuan (bukan laki-laki bentuk wajah laki-laki lebih kotak, sedangkan perempuan (bukan laki-laki) membentuk wajah berbentuk hati serta dagu laki-laki lebih melebar, sedangkan dagu perempuan lebih membulat (Kalansuriya & Dharmaratne, 2015) [5].

Dari banyaknya faktor pembeda yang ada, kumis dan janggut digunakan sebagai fitur khusus menjadi pembeda antara laki-laki dan bukan laki-laki. Fitur khusus ini digunakan karena memiliki ciri-ciri pembeda yang jelas pada wajah dan memiliki nilai yang signifikan untuk dijadikan pembeda antara laki-laki dan bukan laki-laki. Berdasarkan data dari survei, lebih dari 50% laki-laki dengan

umur dari 18 hingga 39 memiliki rambut di wajah di negara Eropa, di daerah Amerika Serikat terdapat 60 hingga 70 persen. Di Jepang, 20% laki-laki memiliki rambut di wajah (Shinichi Terada, 2018) [6]. Adapun adanya keadaan dimana kehadiran rambut kasar pada bukan laki-laki dalam pendistribusiannya seperti laki-laki yang disebut dengan *hirsutisme*. Persentase bukan laki-laki dapat menumbuhkan kumis dan janggut adalah sebesar 5-10% (Sachdeva, 2018) [7].

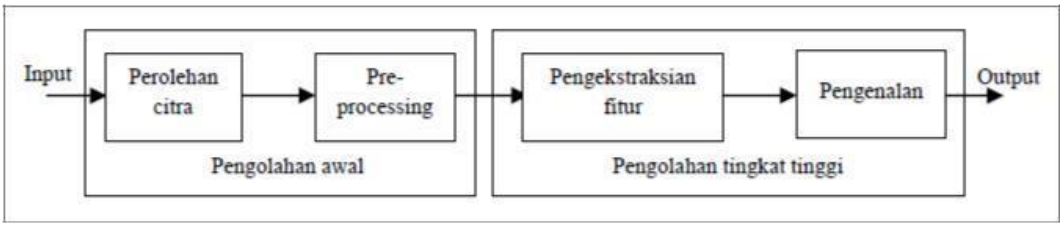
2.2 Visi Komputer

Visi komputer merupakan suatu bidang yang berhubungan dengan pengolahan otomatis suatu citra berbasis komputer untuk mengekstrak dan meinterpretasikan suatu informasi yang didapatkan. Sistem kerja visi komputer adalah dengan cara memproses data citra yang diinput dan menggunakan kombinasi algoritma pengolahan citra serta kecerdasan buatan sehingga dapat menghasilkan suatu informasi dari citra tersebut. Berbagai macam aplikasi dunia nyata menggunakan *computer vision* meliputi:

- *Optical Character Recognition* (OCR) : sering digunakan untuk mengidentifikasi citra huruf untuk kemudian diubah ke dalam bentuk file tulisan, seperti membaca kode pos tulisan tangan pada surat, dan pengenalan otomatis nomor plat atau *Automatic Number Plate Recognition* (ANPR)
- *Medical Imaging* (Pencitraan Medis) : mencatat atau merekam citra *pre-operative* dan *intra-operative* untuk tujuan klinis atau melakukan studi jangka panjang dari morfologi citra bagian tubuh manusia.

- *3D Model Building* : konstruksi otomatis model 3D dari foto udara yang digunakan dalam sistem seperti Bing Maps.
- *Automotive Safety* : mendeteksi rintangan yang tak terduga seperti pejalan kaki di jalan raya.
- *Surveillance* : pemantauan untuk penyusup dan menganalisis lalu lintas jalan raya (Szeliski, 2010) [8].
- *Face Recognition* : identifikasi wajah berdasarkan karakteristik tertentu sesuai dengan informasi yang kita inginkan (Zhao, 2006) [9]

Pengidentifikasian wajah, pengidentifikasian tanda tangan, pengidentifikasian tumor pada suatu citra resonansi magnetik, pengenalan objek pada citra satelit, penempatan sumber daya mineral dari suatu citra, penghasilan gambaran tiga-dimensi dari potongan citra dua dimensi, dan pengenalan suatu kode ZIP, dianggap berada dalam ruang lingkup *computer vision*. Manusia memiliki kemampuan untuk mengenal dan mengklasifikasikan citra, mengidentifikasi citra yang terhalang sebagian pada lingkungan yang memiliki *noise* (gangguan) pada citra, mengidentifikasi objek dengan orientasi dan skala yang berbeda, serta kedalaman persepsi. Proses mengenal dan mengklasifikasikan suatu citra ini dapat pula dilakukan oleh komputer yang dikenal dengan *computer vision*. Proses pengenalan tersebut terlihat pada Gambar 2.1.



Gambar 2.1 Sistem *Computer Vision*

Pengembangan sistem visi komputer hingga saat kini untuk melaksanakan tugas-tugas seperti ini membutuhkan proses kerja yang kompleks. Biasanya untuk setiap aplikasi yang diberikan, keseluruhan tugas tidak dapat dilaksanakan pada sebuah tahapan tunggal. Oleh karena itu, sistem *computer vision* seringkali dibagi ke dalam beberapa tahapan, dan setiap tahapan melaksanakan satu fungsi atau lebih. Sistem *computer vision* tertentu terdiri dari tahapan-tahapan seperti perolehan citra atau sering disebut dengan input citra *preprocessing*, pengekstrasian fitur (mengambil ciri khusus pada suatu citra), penyimpanan objek secara asosiatif, pengaksesan suatu basis pengetahuan, dan pengenalan suatu citra dikelompokkan menjadi kategori tertentu.

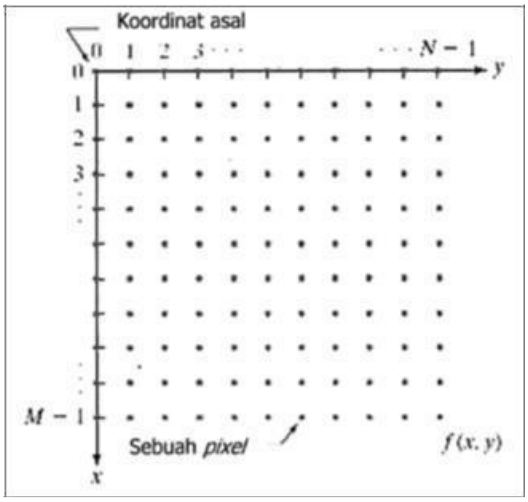
2.3 Citra Digital

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra digital merupakan kumpulan piksel (*picture element*) dengan suatu intensitas tertentu. Resolusi atau dimensi citra merupakan ukuran dari sebuah citra yang dinyatakan dengan *pixel's pixel*. Semakin tinggi resolusi suatu citra, maka akan semakin baik tampilan dari citra digital tersebut. Bagian terkecil dari suatu citra disebut dengan *pixel*. Sedangkan intensitas (kedalaman *bit*) dari masing-masing *pixel*, secara keseluruhan akan menggambarkan terang atau gelapnya citra digital tersebut.

Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optic berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televise, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpanan.

Citra terbagi menjadi dua yaitu citra diam (*still image*) dan citra bergerak (*moving image*). Citra diam adalah citra tunggal yang tidak bergerak. Sedangkan citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun (*sequentiali*) sehingga memberi kesan pada mata sebagai gambar yang bergerak.

Pengolahan citra digital menunjuk pada pemrosesan gambar dua dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital, mengacu pada pemrosesan setiap data dua dimensi. Citra digital merupakan sebuah larik (*array*) yang berisi nilai-nilai real maupun kompleks yang direpresentasikan dengan deretan bit tertentu. Suatu citra dapat didefinisikan sebagai fungsi $f(x,y)$ berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan *amplitude f* di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai *amplitude f* secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital.



Gambar 2.2 Koordinasi Citra Digital (Putra, 2010) [10]

Nilai pada suatu irisan antar baris dan kolom (pada posisi *x*,*y*) disebut dengan *picture elements*, *image elements*, *pels*, atau *pixels*. Istilah terakhir (*pixel*) paling sering digunakan pada citra digital. Citra digital tersusun atas titik-titik yaitu dapat berbentuk persegi panjang dan secara beraturan membentuk baris-baris dan kolom-kolom. Setiap titik memiliki koordinat dan dapat dinyatakan dalam bilangan bulat positif, yaitu 0 atau 1 bergantung pada sistem yang digunakan. Format nilai pixel sama dengan format citra keseluruhan. Pada kebanyakan sistem pencitraan, nilai ini dapat berupa bilangan bulat positif. Format citra digital yang banyak digunakan, yaitu:

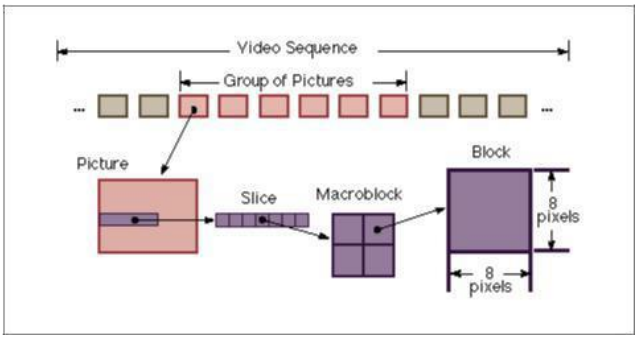
- Citra biner (Monokrom). Citra monokrom atau citra hitam putih merupakan citra yang setiap pikselnya hanya mempunyai dua kemungkinan nilai, seperti *on* dan *off*, disimpan dalam matriks dengan nilai 0 (*off*) dan 1 (*on*).
- Citra skala keabuan (*Grayscale*). Citra *grayscale* dikatakan format citra skala keabuan karena pada umumnya warna yang dipakai adalah warna

hitam sebagai warna minimum dan warna putih sebagai warna maksimumnya, sehingga warna antara keduanya adalah abu-abu.

- Citra berwarna, dimana citra warna terdiri atas 3 layer matriks, yaitu *R-layer*, *G-layer*, *B-layer*. Sistem warna RGB (*Red Green Blue*) menggunakan sistem tampilan grafik kualitas tinggi (*High quality raster graphic*) yaitu mode 24 bit. Setiap komponen warna merah, hijau, biru masing-masing mendapatkan alokasi 8 bit untuk menampilkan warna. Pada sistem warna RGB, setiap *pixel* akan dinyatakan dalam 3 parameter dan bukan nomor warna. Setiap warna mempunyai *range* nilai 00 (angka desimalnya adalah 0) dan *f* (angka desimalnya 255) atau mempunyai nilai derajat keabuan $256 = 2^8$. Dengan demikian, *range* warna yang digunakan adalah (2^8) (2^8) $(2^8) = 2^{24}$ (atau dikenal dengan istilah *true color* pada Windows). Nilai warna yang digunakan merupakan gabungan warna cahaya merah, hijau dan biru.

2.4 Video Digital

Video adalah bentuk penerapan teknologi untuk menangkap, merekam, memproses, menyimpan, dan merekonstruksi kumpulan citra yang saling berurutan. Alan C. Bovik dalam bukunya *Handbook of Image and Video Processing* menjelaskan bahwa video digital merupakan hasil samping dan kuantisasi dari video analog. Secara mendasar, tidak ada perbedaan proses sampling dan kuantisasi antara citra digital dan video digital.



Gambar 2.3 Struktur video digital

(http://www.hongik.edu/~sjpark/mpeg.html)

Dari gambar 2.3, struktur yang menyusun video yaitu (Mukhopadhyay, 2011) [11] :

1. *Video Sequence*, diawali dengan *sequence header*, berisi satu grup gambar atau lebih, diakhiri dengan kode *end-of-sequence*.
2. *Group of Pictures* (*GOP*), *Header* dan serangkaian satu atau lebih gambar yang dimaksudkan untuk memungkinkan akses secara acak menjadi beruntun.
3. *Picture*, Unit pengkodean utama dari urutan video. Gambar terdiri dari tiga matriks segi empat yang mewakili nilai pencahayaan (*Y*) dan dua ninlai krominasi (*Cb* dan *Cr*).
4. *Slice*, satu atau beberapa *macroblocks* bersebalahan. Urutan *macroblocks* dalam *slice* adalah dari kiri ke kanan dan dari atas ke bawah.
5. *Macroblocks*, komponen pencahayaan dan komponen krominasi sesuai dengan urutan blok pada aliran data.

Video digital pada dasarnya adalah sekumpulan citra digital yang disusun secara teratur dan berurutan sehingga menyebabkan efek objek yang ada di dalam

citra digital tersebut bergerak, dikarena adanya citra yang saling berurutan. Video digital terdiri dari beberapa *frame* yang ditampilkan dengan kecepatan tertentu. Setiap *frame* merupakan citra kontinu dan kecepatan untuk menampilkan citra-citra yang ada disebut sebagai *frame rate* dengan satuan *fps* (*frame per second*). Jika *frame rate* pada suatu video digital cukup tinggi, maka video akan terlihat semakin halus, dikarenakan banyakanya citra yang menyusun sebuah video tersebut.

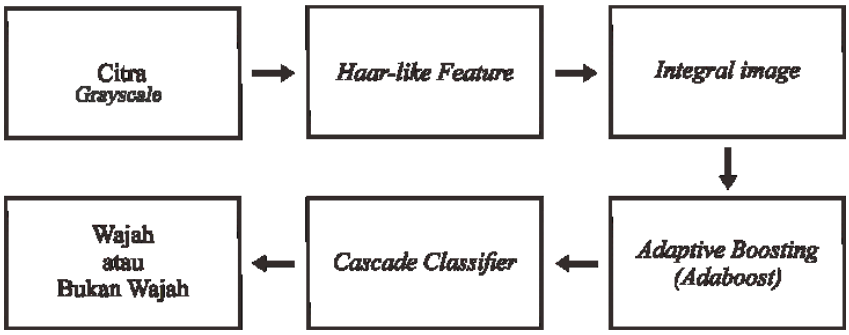
Kualitas suatu video sangat dipengaruhi oleh besarnya nilai karakteristik yang dimiliki oleh sebuah video digital, sehingga sensitifitas mata manusia terhadap video yang dilihat dipengaruhi oleh besarnya nilai-nilai karakteristik dari video itu sendiri. Adapun karakteristik yang dimiliki oleh sebuah video digital adalah :

1. Resolusi. Resolusi atau dimensi *frame* merupakan ukuran sebuah *frame* yang dinyatakan dengan *pixel* x *pixel*. Semakin tinggi resolusi maka semakin baik tampilan video tersebut, namun resolusi yang tinggi membutuhkan jumlah bit yang besar, sehingga memiliki ukuran file yang besar.
2. Kedalaman Bit. Kedalaman bit akan menentukan jumlah bit yang digunakan untuk merepresentasikan tiap piksel pada sebuah frame. Sama halnya dengan resolusi, semakin besar kedalaman bit yang digunakan, maka semakin besar jumlah bit yang dibutuhkan.
3. Laju *Frame*. Laju *frame* merupakan banyaknya jumlah *frame* yang bergerak tiap detik yang kerap dikenal sebagai *frame per second* (fps). Karakteristik ini berkaitan dengan kehalusan gerakan (*smoothness of motion*) sebuah

objek di dalam video. Beberapa nilai standar *frame per second* (fps) yang umum digunakan adalah 30 fps dan 25 fps.

2.5 Metode Viola Jones

Deteksi wajah dilakukan dengan menggunakan metode *Viola - Jones*, ada beberapa proses yang dilakukan sebelum akhirnya akan menghasilkan sebuah output wajah yang terdeteksi pada sebuah citra, proses tersebut dapat dilihat pada Gambar 2.4. Dimana proses-proses tersebut yaitu *Haar-Like Feature*, *Integral image, Adaboost* (*Adaptive Boosting*) dan *Casccade Classifier* (Shulur dkk, 2015) [12].

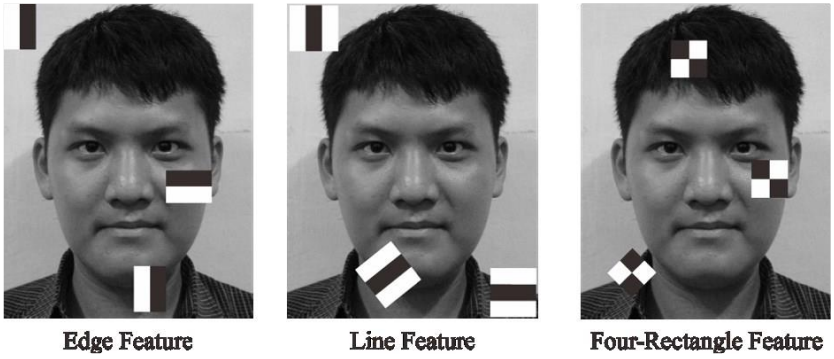


Gambar 2.4 Blok Diagram *Viola-Jones*

2.5.1 Haar-like feature

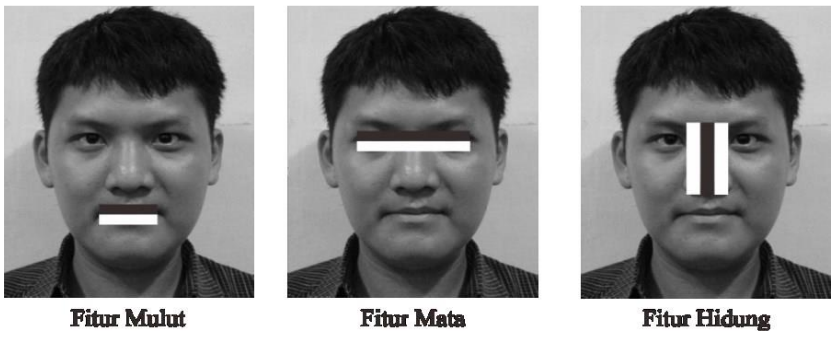
Untuk mendeteksi adanya fitur wajah pada sebuah citra, proses yang dilakukan yaitu memilih fitur *Haar* yang ada pada citra tersebut yang dalam metode *Viola Jones* disebut dengan *Haar-Like feature*. Teknik yang dilakukan yaitu dengan cara mengkotak-kotakkan setiap daerah pada citra mulai dari ujung kiri atas sampai kanan bawah. Proses ini dilakukan untuk mencari apakah ada fitur wajah pada area tersebut. Dalam metode *Viola Jones*, ada beberapa jenis fitur yang bisa digunakan seperti *Edge-feature*, *Line feature*, dan *Four-rectangle feature*. Pada proses pemilihan fitur *Haar*, fitur-fitur tersebut digunakan untuk

mencari fitur wajah seperti mata, hidung, dan mulut. Pada setiap kotak-kotak fitur tersebut terdiri dari beberapa *pixel* pada kotak gelap. Apabila nilai selisih antara daerah terang dengan daerah gelap di atas nilai ambang (*threshold*), maka daerah tersebut dinyatakan memiliki fitur. Proses pemlihan fitur dapat dilihat pada Gambar 2.5.



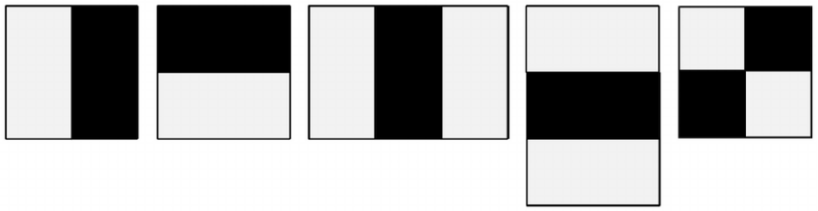
Gambar 2.5 Pemilihan Fitur Wajah

Untuk memilih fitur mata, hidung dan mulut maka digunakan kotak-kotak fitur yang bisa dilihat Gambar 2.6.



Gambar 2.6 Pemilihan Fitur Mulut, Mata dan Hidung

Untuk jenis fitur itu sendiri terbagi atas 3 jenis fitur berdasarkan jumlah persegi panjang (terang dan gelap) yang terdapat di dalamnya yaitu: dua, tiga, empat persegi panjang seperti yang dapat dilihat pada gambar 2.7.



Gambar 2.7 *Haar-Like Feature* (Shulur dkk, 2015) [12]

Cara menghituung nilai dari fitur tersebut dengan (Shulur dkk, 2015) [12] mengurangkan nilai *pixel* pada area putih. Berikut adalah persamaan untuk mendapatkan nilai fitur sesuai dengan jumlah kotak :

B_1 *W* = *Black* (Hitam), *White* (Putih)

Dua kotak : *W-B* (2.2)

Tiga kotak : *W1* + *W2* – *B* (2.3)

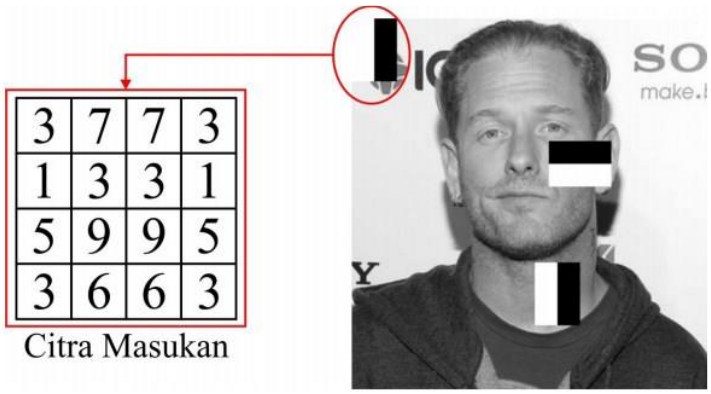
Empat kotak : (*W*1 + *W2*) – (*B1* + *B2*) (2.4)

Untuk mempermudah dan mempercepat proses perhitungan nilai *Haar* pada sebuah citra, metode *Viola Jones* menggunakan sebuah perhitungan yang disebut dengan *Integral Image*.

2.5.2 Integral Image

Integral image sering digunakan pada algoritma untuk pendeteksian wajah. Dengan menggunakan *integral image* proses perhitungan bisa dilakukan hanya dengan satu kali *scan*, memakan waktu yang cepat dan akurat. *Integral image* digunakan menghitung hasil penjumlahan nilai *pixel* pada daerah yang dideteksi oleh fitur *Haar*. Nilai-nilai *pixel* yang akan dihitung merupakan nilai-nilai *pixel* dari sebuah citra masukan yang dilalui oleh fitur *Haar*, pada saat pencarian fitur wajah. Pada setiap jenis fitur yang digunakan, pada setiap kotak-kotaknya terdiri dari

beberapa *pixel*. Apabila ada sebuah masukan yang dilalui oleh fitur oleh fitur *Haar* dapat dilihat pada Gambar 2.8 (Shuhur dkk, 2015) [12].



Gambar 2.8 Nilai *pixel-pixel* pada sebuah fitur

Dari nilai-nilai *pixel* yang didapatkan pada fitur tersebut, maka akan dihitung nilai *integral image* pada fitur tersebut dengan persamaan (2.5).

$$s(x,y) = i(x,y) + s(x,y) + s(x-1,y) – s(x-1,y-1) \quad (2.5)$$

s(x,y) = merupakan nilai hasil penjumlahan dari tiap-tiap *pixel*.

i(x,y) = merupakan nilai intensitas diperoleh dari nilai *pixel* dari citra masukan.

s(x-1,y) = merupakan nilai *pixel* pada sumbu x.

s(x,y-1) = merupakan nilai *pixel* pada sumbu y.

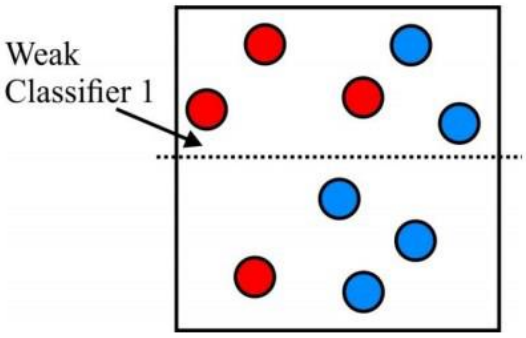
s(x-1,y-1) = merupakan nilai *pixel* diagonal.

2.5.3 Adaboost (Adaptive Boosting)

Adaptive boosting merupakan teknik yang digunakan untuk mengkombinasikan banyak *classifier* lemah untuk membentuk suatu gabungan *classifier* yang lebih baik. Proses dari *adaptive boosting* akan menghasilkan sebuah *classifier* yang kuat dari *classifier* dasar. Satuan dari *classifier* dasar tersebut disebut dengan *weak learner*. Setelah sebelumnya dilakukan pemilihan fitur *Haar*,

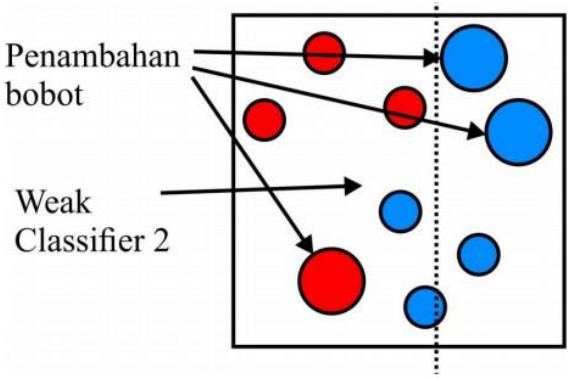
pada proses selanjutnya dalam deteksi wajah *Viola Jones*, dengan menggunakan algoritma *adaboost* fitur pada sebuah citra akan dideteksi kembali. Tujuannya untuk mengetahui apakah ada fitur wajah pada daerah dengan klasifikasi fitur yang lemah. Pada *classifier* lemah akan dilakukan perhitungan dan dibandingkan dengan *classifier* lainnya secara acak. Selanjutnya dilakukan kombinasi atau penggabungan pada *classifier* lemah untuk membentuk suatu kombinasi yang linier.

Pada Gambar 2.9 di bawah ini menunjukkan beberapa *classifier* yang lemah pada sebuah fitur *image*. Lingkaran merah menunjukkan sebuah *classifier* yang lemah sedangkan lingkaran baru menunjuukkan *classifier* kuat. Daerah dengan banyak fitur lemah diklasifikasikan sebagai daerah dengan klasifikasi yang lemah.



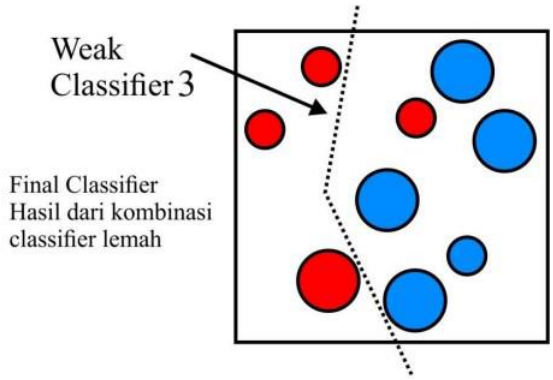
Gambar 2.9 *Classifier* Lemah (Shulur dkk, 2015) [12]

Pada Gambar 2.9 didapatkan beberapa fitur dengan klasifikasi yang lemah maka bobot dari fitur tersebut akan digabungkan untuk meningkatkan bobot dari fitur tersebut agar bisa menjadi fitur dengan *classifier* yang kuat. Hasil dari proses penggabungan *classifier* lemah dengan *classifier* kuat dapat dilihat pada Gambar 2.10



Gambar 2.10 Hasil Kombinasi dari *Classifier* Lemah (Shulur dkk, 2015) [12]

Apabila masih terdapat *weak classifier* pada sebuah fitur setelah dilakukan kombinasi atau penggabungan pada sebuah daerah dengan klasifikasi yang lemah, maka daerah tersebut tetap dianggap sebagai *weak classifier* yang berarti tidak terdapat fitur wajah pada daerah tersebut. Hasil akhir dari penggabungan *classifier* pada algoritma *adaboost*, dapat dilihat pada gambar 2.11.



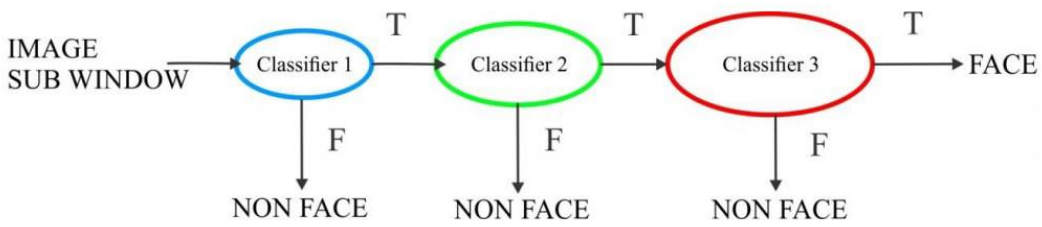
Gambar 2.11 Hasil Kombinasi Dari *Classifier* Lemah (Shulur dkk, 2015) [12]

2.5.4 Cascade Classifier

Cascade classifier melakukan proses dari banyak fitur-fitur mengorganisirnya dengan bentuk klasifikasi bertingkat. Terdapat tiga buah

klasifikasi untuk menentukan apakah ada atau tidak ada fitur wajah pada fitur yang sudah dipilih (Shulur dkk, 2015) [12].

Pada klasifikasi filter pertama, tiap subcitra akan diklasifikasi menggunakan satu fitur. Jika hasil nilai fitur dan filter tidak memenuhi kriteria yang diinginkan, hasil tersebut akan ditolak. Algoritma kemudian bergerak ke *sub window* selanjutnya dan menghitung nilai fitur kembali. Jika didapt hasil sesuai dengan *threshold* yang diinginkan, maka dilanjutkan ke tahap filter selanjutnya. Hingga jumlah *sub window* yang lolos klasifikasi akan berkurang hingga mendekati citra yang dideteksi. Pada Gambar 2.12 di bawah ini merupakan proses rangkaian filter yang dilalui oleh setiap *classifier* (Shulur dkk, 2015) [12].



Gambar 2.12 *Cascade Classifier* (Shulur dkk, 2015)

Pada filter pertama dipilih satu fitur *classifier* dengan presentasi tingkat pendeteksian sebesar 100% dan sekita 50% tingkat kesalahan. Pada filter kedua dipilih lima buah fitur *classifier* dengan persentase tingkat pendeteksian sebesar 100% dan 40% tingkat kesalahan (20% kumulatif). Pada filter ketiga dipilih 20 fitur *classifier* dengan persentasi tingkat pendeteksian sebesar 100% dengan tingkat kesalahan sebesar 10% (2% kumulatif).

Hasil pendeteksian bisa berupa wajah atau bukan wajah. Citra tersebut akan ditandai dengan sebuah *rectangle* pada daerah wajah yang terdeteksi dan apabila

tidak ada wajah terdeteksi maka tidak aka nada *rectangle*, citra tersebut tidak akan ditandai oleh sebuah *rectangle* (Shulur dkk, 2015) [12].

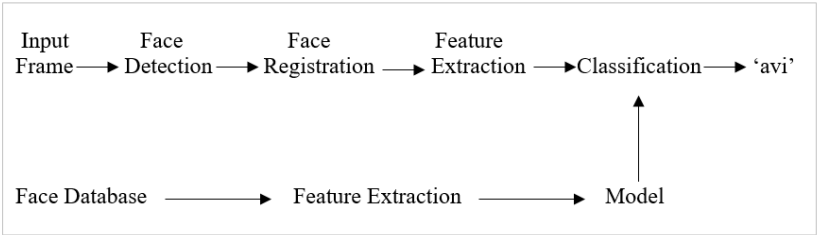
2.6 Face Recognition

Face Recognition adalah proses mengidentifikasi orang dalam gambar atau *frame* video dengan membandingkan tampilan wajah dalam citra yang diambil dengan yang ada pada *database*.

Sebagai sistem biometrik, *face recognition* beroperasi dalam salah satu atau kedua mode: (1) verifikasi wajah (atau otentikasi), dan (2) identifikasi wajah (atau pengenalan).

Verifikasi wajah merupakan sebuah sistem *one-to-one match* yang membandingkan citra wajah *query* dengan citrra wajah pendaftaran yang identitasnya diklaim. Salah satu aplikasi yang khas adalah penggunaan *E-passport* untuk pembersihan imigrasi swalayan.

Identifikasi wajah merupakab sebuah sistem pencocokan *one-to-match* yang membandingkan citra wajah *query* dengan beberapa citra wajah dalam *database* pendaftaran untuk mengaitkan identitas *query* wajah dengan salah satu yang ada di dalam *database*.



Gambar 2.13 *Face Recognition Workflow* (Li, 2011) [13]

2.7 Local Binary Pattern

Local Binary Pattern (LBP) adalah operator tekstur sederhana namun sangat efisien yang melabeli pixel-pixel setiap gambar dengan mengirik linkungan setiap pixel dan menganggap hasilnya sebagai bilangan biner. Karena keserhanaan komputasinya, operator tekstur LBP telah menjad pendekatan popular dalam berbagai apliaksi. Salah satu property dan mungkin merupakan property yang paling penting dari operator LBP dalam aplikasi dunianyata adalah ketahanannya terhadap perubahan skala abu-abu monoton yang disebabkan, misalnya oleh variasi pencahayaan. Properti penting lainnya, seperti yang telah disebutkan sebelumnya yaitu keserderhanaan komputasina, yang memungkinkan untuk menganalisis gambar dalam pengaturan real-time yang menantang (Prado, 2017) [16].

Ide dasar untuk mengembangkan operator LBP adalah bahwa tekstur permukaan dua dimensi dapat dijelaskan oleh dua Langkah komplementer yaitu pola spasial local dan kontras skala abu-abu. Operator asli LBP membentuk label untuk pixel gambar dengan mengesampingkan 3x3 lingkungan setiap pixel dengan nilai tengah dan mempertimbangkan hasilnya sebagai bilangan biner. Histogram dari 28 = 256 tabel ini kemudian dapat digunakan sebagai deskripsi tekstur. Operator ini digunakan Bersama-sama dengan ukuran kontras lokal sederhana yang menyediakan kinerja yang menyediakan kinerja yang sangat baik dalam segmentasi tekstur tanpa pengawasan. Setelah ini, banyak pendekatan terkait telah dikembangkan untuk tekstur dan segmentasi tekstur warna.

Dengan menerapkan konsep *local binary pattern* (LBP) dan di kombinasikan dengan histogram maka dapat mewakili gambar wajah dengan vector data sederhana. Karena LBP adalah pendeskripsi visual maka juga dapat digunakan

untuk tugas pengenalan wajah, seperti yang dapat dilihat dari penjelasan selangkah demi selangkah berikut ini.

2.7.1 Parameter

Local Binary Pattern Histogram menggunakan empat parameter yaitu:

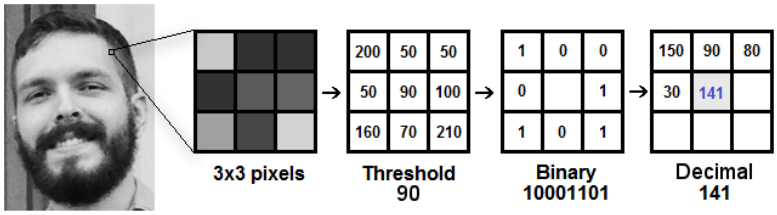
- Radius, digunakan untuk membangun pola biner lokal melingkar dan mewakili radius di sekitar pixel pusat.
- *Neighbors*, jumlah titik sampel yang digunakan untuk membangun pola biner lokal melingkar dan semakin banyak titik sampel yang dimasukkan maka semakin tinggi pula nilai atau biaya komputasi.
- *Grid X*, merupakan jumlah sel dalam arah horizontal. Semakin banyak sel, semakin halus grid, semakin tinggi pula dimensi dari vektor yang dihasilkan.
- *Grid Y*, merupakan jumlah sel dalam arah vertikal. Semakin banyak sel, semakin halus grid, semakin tinggi pula dimensi dari vektor yang dihasilkan.

2.7.2 Melatih Data

Hal berikutnya setelah menentukan atau melaukan pengaturan pada parameter yaitu melatih data dengan menggunakan algoritma pelatihan. Untuk melakukannya, kita perlu menggunakan kumpulan data dengan gambar wajah orang-orang yang ingin kita kenali. Kita perlu juga mengatur ID (memungkinkan nomor atau nama orang tersebut) untuk setiap gambar, gambar orang yang sama harus memiliki ID atau label yang sama, sehingga algoritma akan menggunakan informasi ini untuk mengenali gambar input dan memberi output.

2.7.3 Menerapkan Operasi Local Binary Pattern Histogram

Langkah komputasi pertama dalam *Local Binary Pattern Histogram* (LBPH) adalah untuk membuat gambar (*intermediate image*) yang mendeskripsikan gambar asli dengan cara yang lebih baik, dengan cara melakukan sorotan (*highlight*) pada karakteristik wajah. Untuk melakukannya, algoritma menggunakan konsep jendela geser (*sliding window*), berdasarkan parameter *radius* dan *neighbors*.



Gambar 2.14 Prosedur Operasi Algoritma LBPH

(https://cdn-images-1.medium.com/max/1600/1*J16_DKuSmAH3WDdqWKeNA.png)

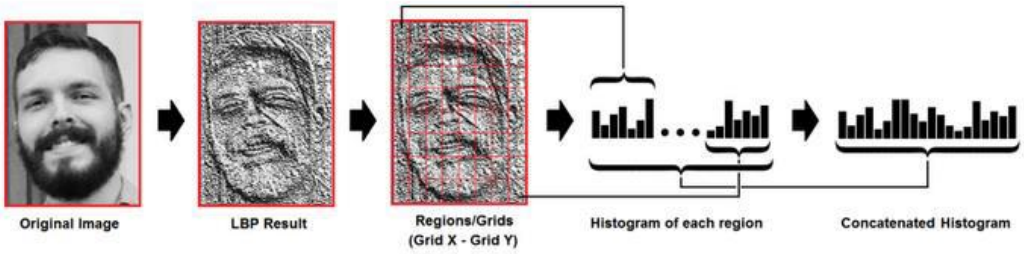
Berdasarkan Gambar 2.14 operasi LBPH (*Local Binary Pattern Histogram*) dapat dibagi sebagai berikut :

- Misalnya kita memiliki gambar wajah dengan skala abu-abu (*grayscale*).
- Mendapatkan bagian dari gambar tersebut dengan jendela 3x3 *pixel*.
- Dapat juga direpresentasikan sebagai matriks 3x3 yang berisi intensitas setiap *pixel* (0-255).
- Mengambil nilai pusat dari matriks yang akan digunakan sebagai ambang batas.

- Nilai ini akan digunakan sebagai nilai baru dari *neighbor* yang telah diatur sebelumnya.
- Untuk setiap *neighbor* dari nilai pusat (ambang), ditetapkan nilai biner baru. Dalam hal ini yaitu 1 untuk nilai yang sama atau lebih tinggi dari ambang dan 0 untuk nilai yang lebih rendah dari ambang (*threshold*).
- Sekarang, matriks hanya akan berisi nilai-nilai biner (mengabaikan nilai pusat). Masing-masing nilai biner perlu digabungkan dari setiap posisi dan garis matriks.
- Kemudian, kita mengubah nilai biner ini menjadi nilai decimal dan mengaturnya ke nilai pusat matriks yang sebenarnya adalah sebuah *pixel* dari gambar aslinya.
- Pada akhirnya prosedur ini (prosedur LBP), kita memiliki gambar baru yang memiliki karakteristik yang lebih baik dibandingkan gambar asli.

2.7.4 Ekstraksi *Histogra*m

Dengan menggunakan gambar yang didapatkan pada langkah sebelumnya, dengan menggunakan parameter grid X dan grid Y yang telah diatur sebelumnya maka gambar tersebut dibagi menjadi beberapa bagian, seperti dapat dilihat pada gambar berikut.



Gambar 2.15 Pembagian gambar oleh parameter grid X & Y

(https://cdn-images-1.medium.com/max/1600/1*-cyqWPcas3CXp4O2O7xPpg.png)

Berdasarkan Gambar 2.15, kita dapat mengekstrak *histogram* dari setiap bagian sebagai berikut:

- Karena gambar yang ada adalah dalam skala abu-abu (*grayscale*), setiap *histogram* (dari setiap kotak) hanya akan berisi 256 posisi (0-255) yang mewakili kemunculan setiap intensitas *pixel*.
- Kemudian perlu digabungkan setiap *histogram* untuk mendapatkan *histogram* baru yang lebih besar. Misalkan terdapat 8x8 grid, maka kita akan memiliki 8x8x256 = 16.384 posisi dalam *histogram* akhir. *Histogram* akhir mewakili karakteristik gambar dari gambar asli (*original*).

2.7.5 Melakukan Pengenalan Wajah (*Face Recognition*)

Pada langkah ini, algoritma telah dilatih (*train*). Setiap *histogram* yang telah dibuat digunakan untuk mempresentasikan setiap gambar (*image*) yang berasal dari dataset gambar latih (*training image*). Kemudian saat diberikan gambar masukan, maka akan dilakukan lagi langkah-langkah seperti di atas pada gambar masukan tersebut untuk membuat *histogram* yang mewakili gambar.

Jadi untuk menemukan gambar yang cocok dengan gambar masukan, hanya perlu membandingkan dua *histogram* dan mengembalikan gambar dengan *histogram* terdekat. Untuk membandingkan dua *histogram* (jarak antara dua *histogram*) dapat dilakukan dengan berbagai pendekatan misalnya dengan menggunakan jarak *Euclidean*, *chi-square*, nilai absolut dan lain-lain.

Jadi output dari algoritma adalah ID dari gambar dengan *histogram* terdekat. Algoritma juga harus menghitung jarak (*distance*) yang dapat digunakan sebagai pengukuran kepercayaan (*confidence*). Yang dapat digunakan bersamaan dengan nilai ambang (*threshold*) untuk memperkirakan secara otomatis bila algoritma telah mengenali gambar dengan benar.

2.8 Python

Pythonn merupakan bahasa pemrograman yang menggunakan metode pemrograman berorientasi objek (OOP) dengan manajemen memori otomatis (*pointer*). Program yang ditulis menggunakan Python dapat dijalankan di hamper semua sistem operasi Linux/Unix, Microsoft Windows, Mac OS, Android, Java, Virtual Machine, Symbian OS, Amiga, Raspberry Pi, dan Palm (python.org) [17].

2.9 OpenCV

OpenCV (*Open Source Computer Vision Library*), adalah sebuah *library open source* yang dikembangkan oleh intel yang berfokus untuk *computer vision*. Banyak algoritma yang ditawarkan oleh OpenCV yang dapat kita gunakan untuk membuat sebuah aplikasi yang berfokus pada *computer vision*. Algoritma-algoritma tersebut dapat digunakan untuk mendeteksi dan mengenali wajah atau

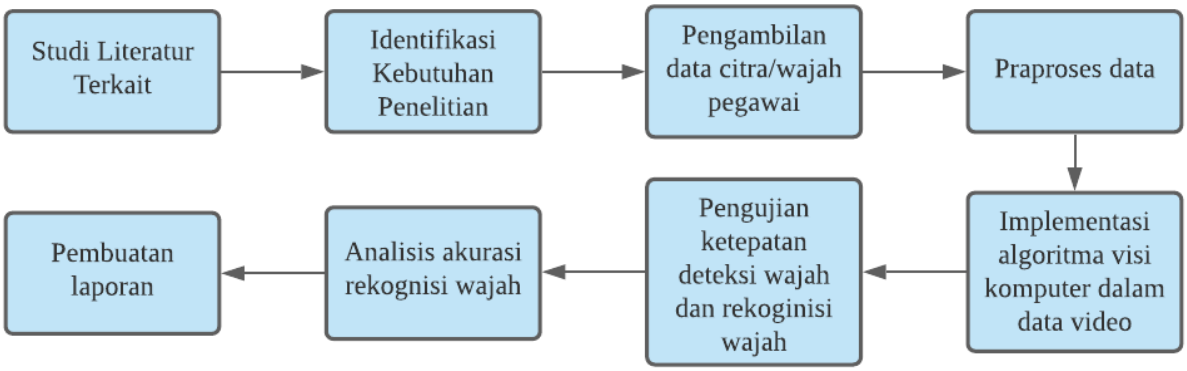
objek, mengklasifikasikan gerakan manusia dalam video, mengikuti pergerakan kamera, mengikuti objek yang bergerak, mengekstrak model 3D dari suatu objek, menggabungkan citra untuk mendapatkan citra yang beresolusi tinggi, mencari gambar yang mirip dalam *database*, menghilangkan efek mata merah dari citra hasil tangkapan kamera *flash* dan masih banyak lagi. Pustaka ini memiliki *interface* untuk C++, C, Python, Java dan MATLAB dan juga mendukung sistem operasi *Windows*, *Linux, Android*, dan *Mac OS*. OpenCV sangatlah handal digunakan pada *realtime vision application* (opencv.org) [18].

BAB III

METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Penelitian ini dilakukan dengan melalui beberapa tahap sebagaimana ditunjukkan pada Gambar 3.1



Gambar 3.1 Tahapan penilitian

Tahapan pada **Gambar 3.1** dapat dijelaskan sebagai berikut:

1. Studi literatur terkait dilkakukan untuk mendapatkan pengetahuan dan dokumentasi tentang penilitan-penilitian sebelumnya serta teori-teori yang terkait deteksi wajah atau rekognisi wajah dan pengidentifikasian identitas wajah pegawai yang terdeteksi menggunakan algoritma Computer Vision. Pada tahap ini juga dilakukan dokumentasi hasil penelitian-penelitian sebelumnya.
2. identifikasi kebutuhan penelitian. Pada tahap ini, dilakukan penetapan berbagai kebutukan penelitian dan disiapkan untuk menunjang penilitian.
3. Setelah menyiapkan kebutuhan penilitian, pengambilan data yang berupa citra wajah pegawai sudah dapat di mulai untuk bahan pengujian sistem.

4. Praproses data, pada data citra yang telah disiapkan dilakukan praproses berupa grayscalling, resizing, cropping. Data yang terproses akan di gunakan untuk pelatihan model untuk rekognisi wajah
5. Implementasi algoritma Computer Vision dalam data video. Data video yang sudah dianalisis tadi kemudian diimplementasikan pada algoritma sistem. Penelitian ini menggunakan algoritma atau metode Viola-Jones dan Local Binary Pattern Histogram (LBPH). Untuk deteksi wajah dan mata menggunakan metode Viola-Jones, LBPH digunakan untuk membuat model dan pengklasifikasian wajah pegawai.
6. Selanjutnya dilakukan uji coba ketepatan sistem untuk deteksi wajah dan rekognisi wajah. Wajah yang terdeteksi diberikan tanda persegi panjang (bounding box) berwarna hijau agar memberikan tanda bahwa wajah berhasil di deteksi dan wajah terdeteksi akan di berikan label berupa nama pegawai atau "tidak dikenali" jika tidak masuk dalam data latih.
7. Selanjutnya menganalisis akurasi rekognisi wajah. Proses perhitungan dilakukan dengan menjumlahkan wajah pegawai yang berhasil direkognisi, wajah pegawai yang tidak direkognisi, wajah bukan pegawai yang salah

direkognisi sebagai wajah pegawai, dan juga wajah bukan pegawai yang tidak direkognisi.

8. setelah didapatkan hasil yang akurat, akan dilakukan penyusunan laporan sebagai dokumentasi proses dan hasil penelitian yang telah dilakukan.

Gambar 3.2 Gambaran umum penelitian

Dari Gambar 3.2 dapat dilihat gambaran umum penelitian ini yaitu pengambilan data citra wajah pegawai yang kemudian digunakan untuk pelatihan model rekognisi wajah, setelah pembuatan model, maka diimplementasikanlah algoritma pada input video sistem tersebut. Implementasi algoritma ini akan mengeluarkan output berupa video yang telah mendeteksi dan menampilkan nama dari pegawai yang sedang dideteksi oleh sistem.

3.2 Waktu dan Lokasi Penelitian

Penelitian dimulai pada bulan Desember 2017 dengan lokasi di Laboratorium Kecerdasan Buatan Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

3.3 Instrumen Penelitian

Instrumen penelitian yang digunakan pada penelitian ini adalah sebagai berikut.

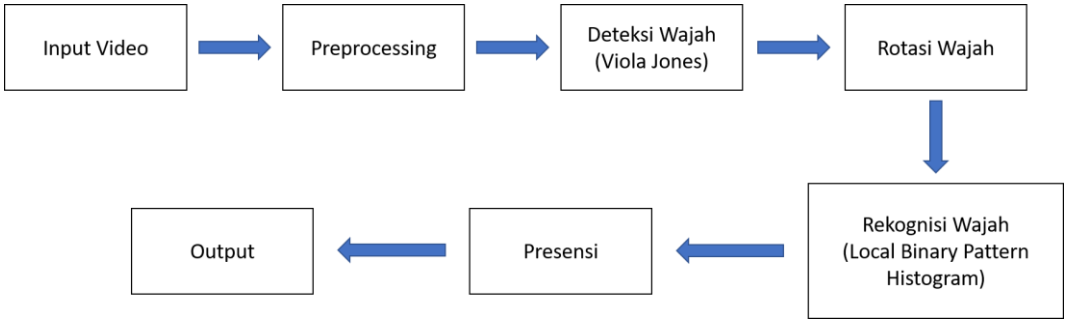
1. Software
    a. Windows 10 64 bit
    b. Python
    c. Pycharm IDE
    d. XAMPP
    e. OpenCV
    f. Tkinter
    g. SQLAlchemy
    h. Reportlab
2. Hardware
    a. Asus X450LD Core i5 RAM 8 GB HDD 500 GB
    b. Webcam Logitech C922 Pro

3.4 Teknik Pengambilan Data

Pengambilan data dilakukan dengan menempatkan kamera 0,55 meter di depan pegawai dengan kondisi wajah dan mata tertangkap jelas dikamera. Untuk data training digunakan 6 citra wajah pegawai dimana setiap pegawai diambil 30 frame citra. Sehingga total data training yang digunakan berjumlah 180 citra wajah. Untuk data testing diambil secara realtime dengan mencocokkan citra yang tertangkap kamera dengan citra yang telah di training.

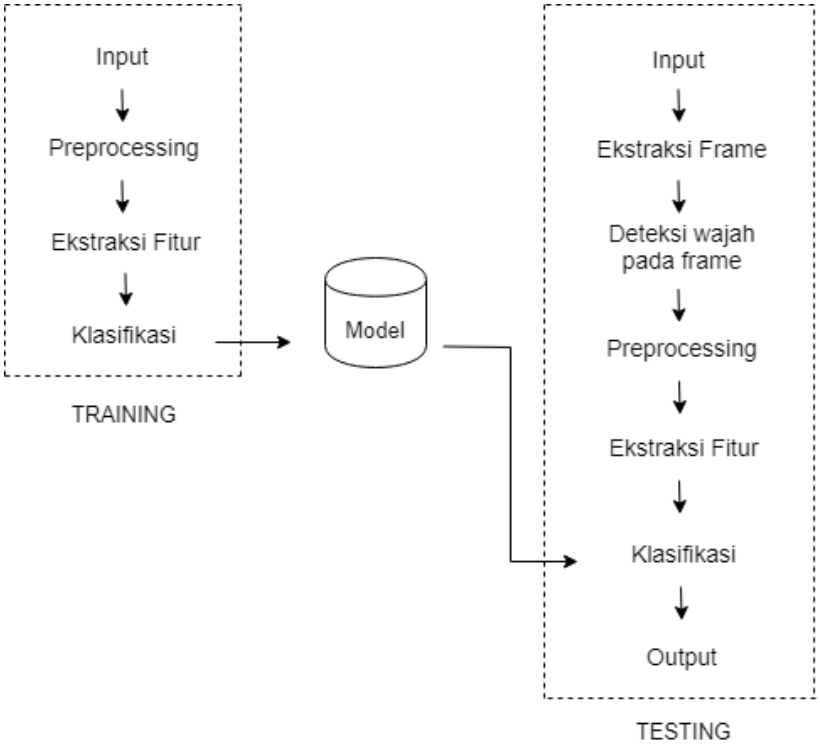
3.5 Perancangan Implementasi Sistem

Secara garis besar, perancangan sistem untuk presensi pegawai menggunakan computer vision terdapat beberapa proses sebagaimana di tunjukkan oleh gambar 3.3.



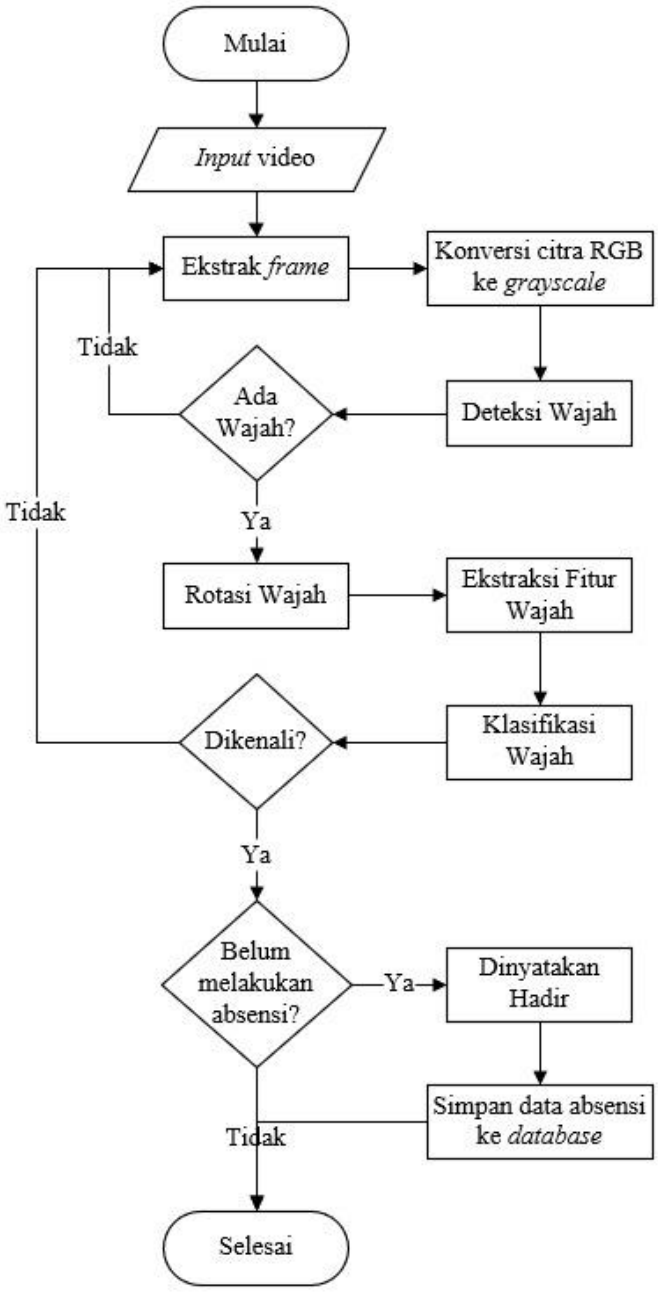
Gambar 3.0.3 Blok diagram perancangan sistem

Dalam perancangan sistem untuk mendeteksi wajah dan mengenali wajah terdapat proses training dan proses testing sebagaimana di tunjukkan gambar 3.4



Gambar 3.4 diagram perancangan sistem

Terdapat juga flowchart alur perancangan dari awal hingga akhir sistem sebagaimana ditunjukkan gambar 3.5.



Gambar 3.5 Flowchart alur perancangan sistem

3.5.1 Proses Training

A. Input Citra Latih

Hal pertama yang harus dilakukan dalam proses pelatihan yaitu mempersiapkan data citra yang akan digunakan sebagai data latih. Input data citra berupa input video secara langsung untuk setiap pegawai menggunakan webcam yang telah disediakan. Input video akan tetap berlangsung sampai jumlah data yang didapatkan mencapai 30 citra wajah. Untuk mendapatkan citra wajah pada saat input data video akan dilakukan ekstraksi frame citra latih dimana frame yand di ekstrak akan dilakukan proses mengubah format citra menjadi grayscale, deteksi wajah dan rotasi wajah sehingga di dapatkan citra data yang diinginkan. Data latih tersebut akan diberikan label pada nama filenya. Jumlah data citra latih yang digunakan sebanyak 30 citra untuk setiap pegawai.

Gambar 3.6 contoh gambar data latih

B. Preprocessing

Setelah melakukan ekstraksi frame, pemrosesan dilanjutkan ke tahap preprocessing untuk persiapan sebelum proses selanjutnya agar dapat digunakan pada metode berikutnya. Proses yang dilakukan adalah mengubah citra RGB

menjadi grayscale. Proses ini dilakukan dengan menghitung nilai rata-rata setiap channel yaitu Red, Green, dan Blue. Kemudian nilai rata-rata tersebut merupakan niali grayscale yang akan digunakan dalam setiap pixel. Untuk menghitung nilai grayscale dari setiap pixel dapat dilakukan dnegan menggunakan persamaan:

$$GS = \frac{R + G + B}{3} \qquad (3.1)$$

Keterangan:

- GS = nilai derajat keabuan (Grayscale).
- R = nilai channer red.
- G = nilai channel green.
- B = nilai channel blue.

Misalkan terdapat sebua matriks $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ dimana nalai dari pixel I merupakan suatu vector tiga dimensi yaitu i = [11, 27, 112]. Nilai dari grayscale dari pixel i adalah:

$$i = \frac{11 + 27 + 112}{3}$$

$$i = \frac{150}{3}$$

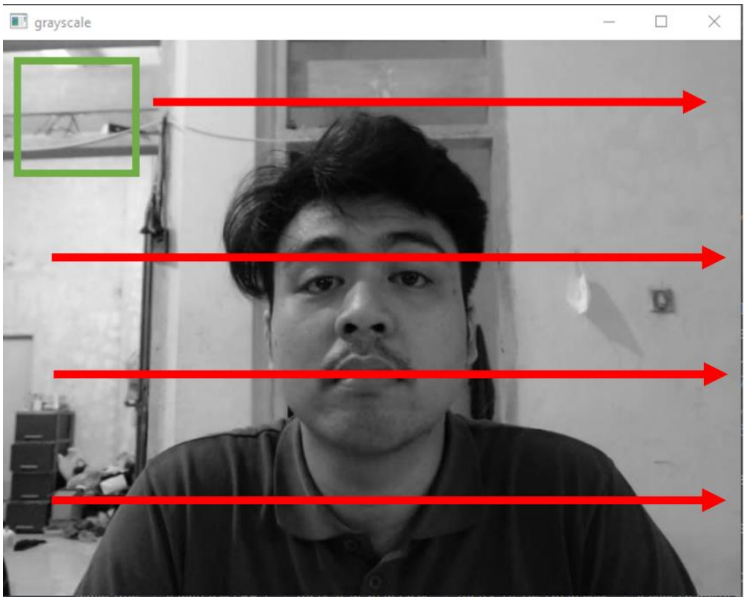
$$i = 50$$

Pada gambar 3.7 dapat dilihat contoh keluaran ekstraksi frame citra RGB yang telah dirubah menjadi grayscale.

Gambar 3.7 contoh gambar grayscale

C. Pendeteksian Wajah (Viola Jones)

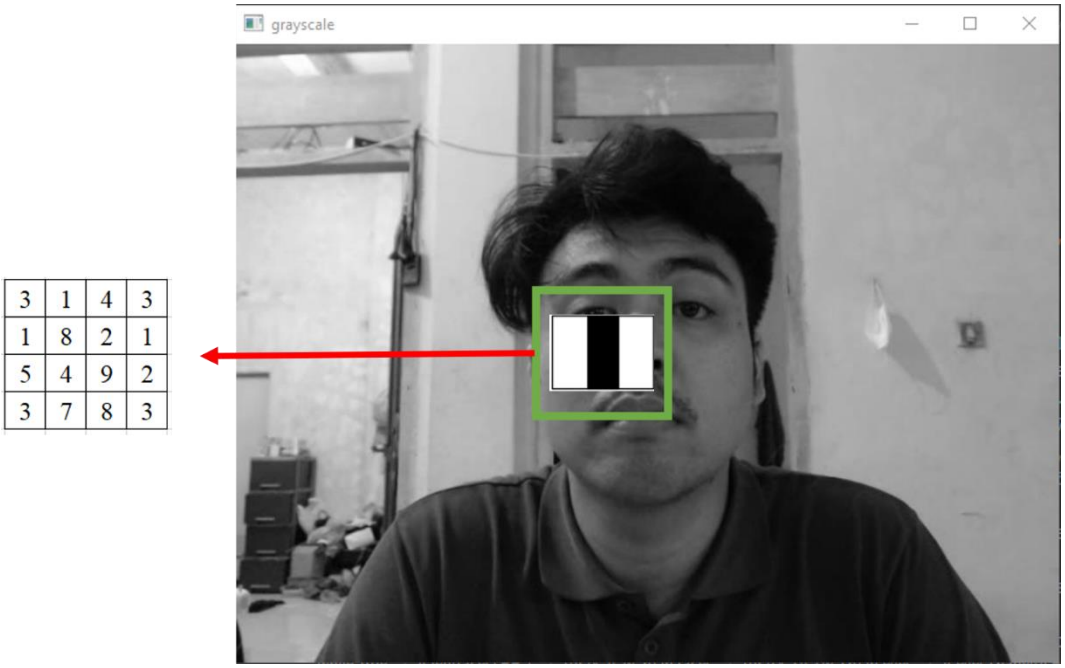
Setelah dilakukan proses grayscale pada citra, selanjutnya akan dilakukan deteksi wajah menggunakan metode viola-jones dengan menggunakan OpenCV yang di program menggunakan Bahasa python. Citra yang telah di rubah menjadi citra grayscale akan menjadi input pada algoritma deteksi wajah viola-jones, proses pertama yang akan dilakukan dalam metode viola-jones yaitu proses pencarian fitur wajah yang disebut dengan haar-like feature, proses ini dilakukan dengan menyusuri tiap pixel ke pixel dari ujung kiri atas hingga ke ujung kanan bawah pada citra diproses, dimana proses ini disebut dengan sliding window, ukuran sliding window yang digunakan yaitu 30x30 pixel, proses dari sliding window dapat dilihat pada gambar 3.8



Gambar 3.8 proses sliding window

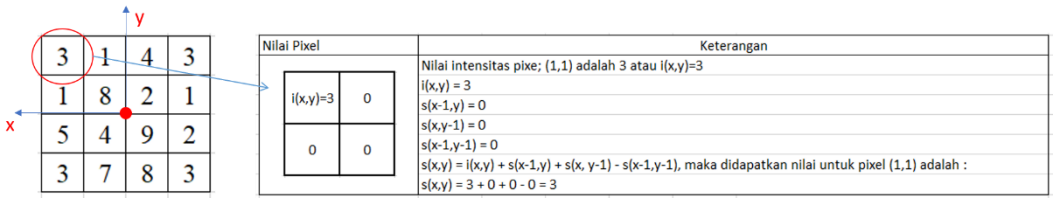
Didalam proses sliding window terdapat proses integral image, proses ini dilakukan untuk menghitung hasil penjumlahan niali pixel yang terdapat pada sliding window yang nantinya akan digunakan untuk pencarian haar-like features

didalam sliding window tersebut. Penggunaan sliding window ini bertujuan untuk mengurangi waktu proses dalam perhitungan jumlah nilai pixel pada fitur yang akan dicari. Misalkan sliding window yang digunakan dalam pencarian fitur wajah berukuran 4x4 pixel, maka sliding window tersebut akan mengambil nilai dari wilayah pixel yang tersebut untuk dilakukan proses integral image yang dapat kita lihat pada gambar 3.9.



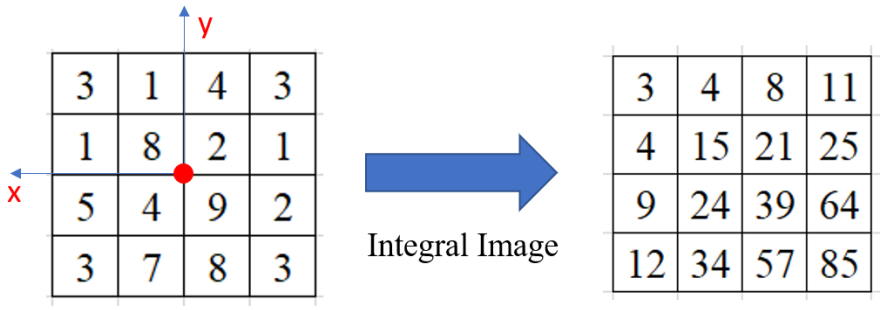
Gambar 3.9 nilaipixel pada sliding window

Berdasarkan nilai pixel pada sliding window pada gambar 3.9, selanjutnya dilakukan perhitungan integral image dengan menggunakan persamaan 2.5. Proses perhitungan integral image yang akan dihitung pertama kali ada 2x2 pixel yang terletak pada sisi kiri, yang dapat dilihat pada gambar 3.10



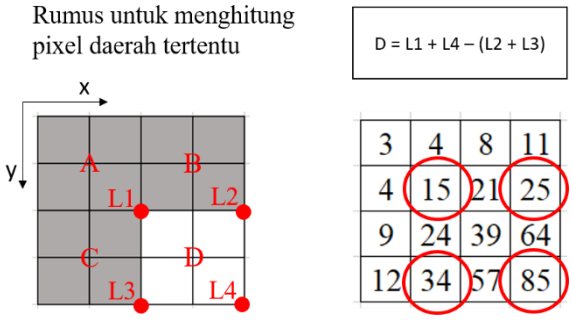
Gambar 3.10 Proses perhitungan integral image

Proses pada gambar 3.10 akan diulang hingga nilai integral image pada area sliding window didapatkan semua dan dilanjutkan ke sisi lain dari sliding window tersebut sehingga didapatkan nilai seperti pada gambar 3.11



Gambar 3.11 Nilai Integral Image pada sliding window

Setelah didapatkan integral image pada daerah tersebut akan dilakukan perhitungan pixel pada area tertentu untuk menemukan haar-like feature dengan cara seperti pada gambar 3.12



Gambar 3.12 Proses perhitungan daerah tertentu

Berdasarkan gambar 3.12, Misalnya daerah tertentu yang akan dihitung ada daerah D, maka nilai yang di ambil ada L1, L2, L3, dan L4, sehingga di dapatkan hasil perhitungan seperti pada gambar 3.13

L1 = 15, L2 =25, L3=34, L4=85

D = 15 + 85 – (25 +34)
= 41

Gambar 3.13 Perhitungan daerah D pada sliding window

Nilai yang didapatkan dari hasil perhitungan daerah tertentu seperti gambar 3.13 selanjutnya akan dibedakan antara daerah gelap dan terang, setelah itu nilai dari pixel bagian gelap dan terang akan dikurangkan, apabila nilai hasil pengurangan pixel gelap dan terang diatas nilai ambang batas maka citra didalam sliding window tersebut memiliki haar-like feature yang ingin dideteksi yang dimana dalam penelitian ini merupakan fitur wajah.

setelah pendetekesian fitur selesai dilakukan, maka akan dihasilkan banyak sekali fitur pada gambar tersebut, fitur yang ditemukan mungkin saja merupakan fitur yang menandakan bukan wajah atau disebut juga fitur lemah, untuk menghindari hal ini maka dilakukan adaptive boosting, adaptive boosting akan mendata kembali semua fitur tersebut untuk mengetahui fitur tersebut merupakan fitur wajah atau bukan, proses dari adaptive boosting ini akan menghasilkan sebuah classifier yang kuat dari classifier dasar, satuan dari classifier dasar tersebut disebut dengan weak learner. Untuk menghasilkan classifier yang kuat dilakukan proses penggabungan classifier lemah, apabila terdapat sebuah daerah pada gambar memiliki beberapa classifier lemah classifier tersebut akan digabungkan untuk mendapatkan classifier kuat apabila setelah penggabungan classifier tersebut telah

dilakukan dan masih terdapat classifier lemah pada daerah tersebut maka daerah tersebut tetap dianggap classifier lemah yang berarti tidak ada fitur wajah di daerah tersebut.

Dalam cascade classifier tersebut semua classifier kuat akan yang telah didapatkan akan dibagi menjadi beberapa stage, dalam setiap stage terdapat beberapa classifier kuat, yang akan mentukan apakah gambar inputan tersebut adalah wajah atau bukan, yang akan menentukan apakah gambar intputan tersebut adalah wajah atau bukan, jika pada stage pertama gambar inputan ditetapkan bukan wajah, maka proses akan berakhir untuk sliding window untuk daerah tersebut dan akan di lanjutkan ke sliding window berikutnya, dan jika sebaliknya maka gambar inputan akan diteruskan ke stage selanjutnya hingga semua stage habis. Jika citra pada sliding window tersebut berhasil melewait semua stage dari maka dapat disimpulkan bahwa didalam sliding window tersebut terdapat wajah. Berikut contoh keluaran dari hasil deteksi wajah yang diberikan bounding box dengan diberikan koordinat x, y, w, dan h untuk penggambaran bounding box pada citra yang ingin dideteksi sebagaimana ditunjukkan pada gambar 3.14

Gambar 3.14 contoh pendeteksian wajah

D. Rotasi Gambar

Setelah mendapatkan koordinat area wajah, proses selanjutnya adalah untuk melakukan rotasi gambar untuk menyajajarkan posisi mata dimana proses pertama adalah melakukan deteksi mata pada area koordinat wajah yang telah terdeteksi dengan menggunakan metode viola-jones.

Setelah mendeteksi dua mata, proses berikutnya adalah mengetahui titik tengah dari kedua mata tersebut. proses untuk mencari titik tengah kedua mata adalah dengan cara menjumlahkan koordinat x dengan setengah nilai dari w, dan juga koordinat y dengan setengah dari nilai h, dimana hasil dari keduanya akan di gabungkan untuk mendapatkan koordinat titik tengah mata untuk kedua mata. Proses berikutnya adalah untuk mencari jarak kedua titik tengah mata berdasarkan sudut siku-siku segitika yang di buat dari kedua titik tengah mata tersebut yang digunukan untuk mencari sudut rotasi yang harus diterapkan pada citra wajah.

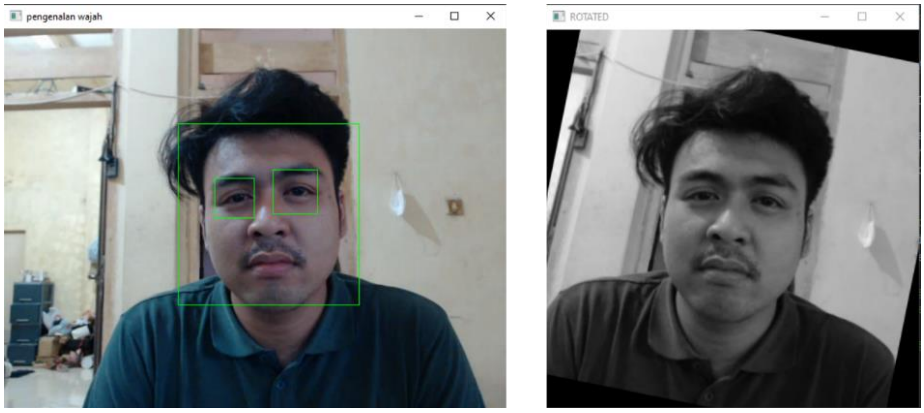
Setelah mendapatkan besar sudut rotasi yang akan digunakan, selanjutnya akan dilakukan proses rotasi yang mana menggunakan transformasi *matrix*. OpenCV memberikan kemampuan untuk menentukan titik tengah rotasi dari gambar dengan factor skala untuk melakukan resize pada citra yang terkait.

Pada fungsi getRotationMatrix2D() memiliki beberapa argument dimana fungsi dari setiap argument tersebut berupa:

- Center: titik tengah rotasi dari citra inputan
- Angle: sudut rotasi gambar dalam satuan derajat
- Scale: factor skala yang digunakan untuk memperbesar atau memperkecil gambar berdasarkan nilai yang diberikan

Pada fungsi tersebut jika nilai angle bernilai positif maka gambar akan dirotasi berlawanan dengan arah jarum jam, jika nilai angle bernilai negatif maka gambar akan dirotasi searah dengan jarum jam.

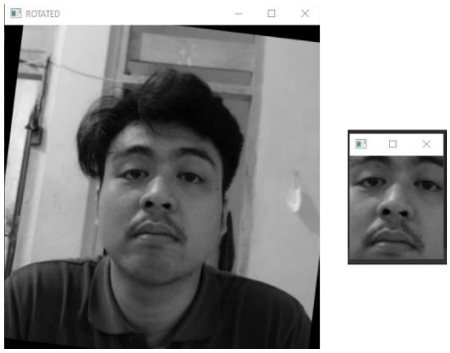
Proses berikutnya melukukan rotasi menggunakan matrix rotasi yang telah dibuat dengan getRotationMatrix2D() dengan menggunakan fungsi warpAffine() yang telah disediakan oleh opencv. Dengan menggunakan fungsi tersebut kita akan mendapatkan hasil rotasi gambar seperti gambar 3.15.



Gambar 3.15 Contoh rotasi gambar

E. Isolasi wajah

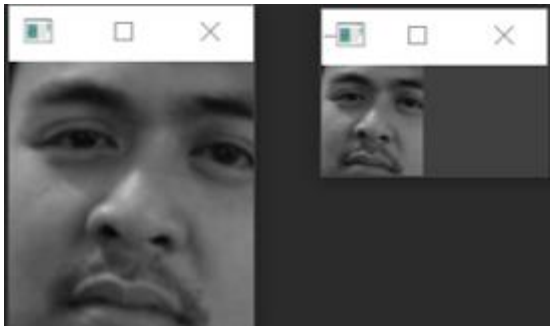
Setelah didapatkan citra ekstraksi fitur wajah yang telah di rotasi, akan dilakukan proses isolasi wajah untuk mengilangkan fitur-fitur selain wajah pada citra, proses ini dilakukan dengan melakukan penjumlahan dan pengurangan pada koordinat wajah pada wajah yang terdeteksi setelah dilakukan rotasi. Contoh dari isolasi wajah dapat dilihat pada gambar 3.16



Gambar 3.16 contoh isolasi wajah

F. Resize gambar

Setelah proses isolasi wajah dilakukan proses resize yang bertujuan untuk menyamakan ukuran wajah yang telah diisolasi, karena pada tahap deteksi dan isolasi wajah ukuran wajah yang dihasilkan berbeda-beda. Pada penilitian ini semua data wajah di-resize menjadi ukuran 54 x 59 pixel seperti pada gambar 3.17.

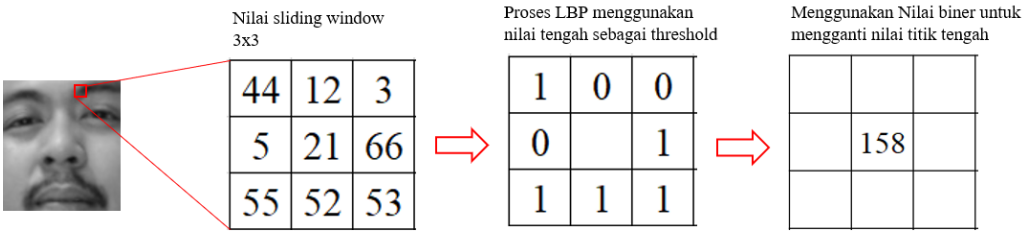


Gambar 3.17 contoh resize gambar

G. Pembuatan model training LBPH

Pada proses pembuatan model data training LBPH menggunakan citra wajah yang telah di resize akan diimplementasikan metode LBP untuk mengekstraksi fitur dari citra wajah yang telah dideteksi dimana akan diterapkan sliding window berukuran 3x3 untuk melakukan metode LBP di setiap pixel pada gambar yang terdeteksi. Kemudian menggunakan nilai tengah dari sliding window tersebut sebagai threshold pada metode LBP dimana kita pada metode tersebut kita

akan mengganti nilai-nilai disekitar titik tengah menjadi biner, dimana jika nilai di sekitar titik tengah berada dibawah threshold nilai disekitar titik tengah akan diubah menjadi 0, sedangkan jika nilai disekitar titik tengah tersebut sama atau lebih dari threshold nilai disekitar titik tengah tersebut akan diganti dengan nilai 1, dimana proses yang telah dijelaskan dapat dilihat pada gambar 3.18.

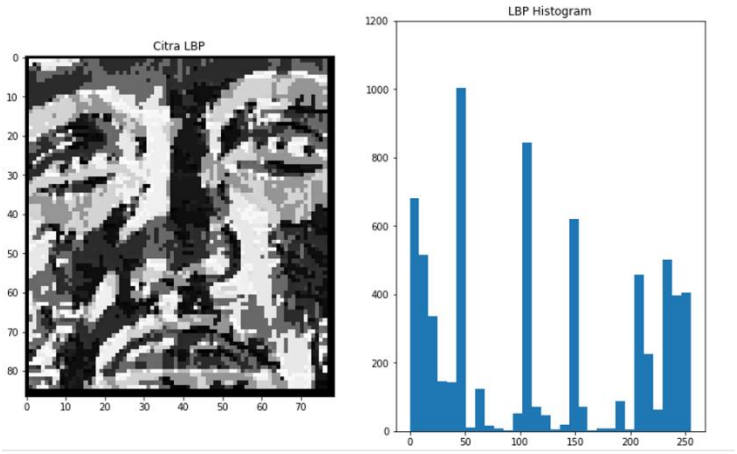


Gambar 3.18 gambaran metode ekstraksi fitur LBP

Proses pada gambar 3.18 akan dilakukan untuk setiap bagian citra yang diinput sehingga dapat menghasilkan gambar lbp seperti pada gambar 3.19.

Gambar 3.19 Contoh hasil proses metode LBP

Kemudian pada gambar tersebut akan dilakukan ekstraksi nilai histrogramnya untuk digunakan sebagai data latih pembuatan model LBPH dimana nilai histogram dari gambar yang telah di emplementasikan metode LBP dimana contoh hasil pengekstraksian histogram dapat dilihat pada gambar 3.20.



Gambar 3.20 hasil ekstraksi histogram pada citra LBP

Hasil proses tersebut dikumpulkan kedalam *file* model yang akan digunakan sebagai acuan untuk melakukan rekognisi wajah pada saat pengujian.

3.5.2 Proses Testing

A. *Input* Frame Video

Pada proses *testing*, *input* yang digunakan adalah *frame video* dari *webcam* yang terhubung ke laptop. *Input* tersebut memiliki resolusi awal 1920x1080 *pixel* dengan *frame rate* 30 fps. Untuk menampilkan *video stream* ke sistem digunakan *library* OpenCV.

B. Preprocessing

Frame yang diperoleh dari *webcam* akan diproses melalui tahap preprocessing yaitu dengan mengubah warna *frame* dari RGB menjadi *grayscale*.

C. Pendeteksian Wajah

Setelah melakukan proses preprocessing, akan dilakukan proses pendeteksian wajah dengan menggunakan metode *Viola-Jones*.

D. Rotasi Gambar

Proses rotasi gambar dilakukan dengan mendeteksi mata pada frame. Setelah kedua mata terdeteksi, akan dilakukan penghitungan titik tengah yang berada di antara kedua mata tersebut. Titik tersebut akan digunakan pada penghitungan matrix rotasi dari frame sehingga kedua mata akan tampak sejajar secara horizontal. Adapun langkah-langkah merotasi wajah adalah sebagai berikut.

a) Mendeteksi mata pada citra.
b) Mengekstrak mata kiri dan kanan.
c) Menghitung titik tengah mata.
d) Menghitung sudut rotasi dengan aturan cosinus.
e) Merotasi citra sesuai sudut rotasi.

E. Ekstraksi Fitur

Melakukan ekstraksi fitur wajah dengan melakukan pengisolasian citra wajah agar tidak terdapat gangguan dari background. Dengan cara melakukan penyempitan region wajah yang terdeteksi.

Metode ekstraksi fitur yang digunakan adalah Local Binary Pattern Histogram (LBPH). Local Binary Pattern adalah operator tekstur sederhana namun sangat efisien untuk melabeli pixel-pixel setiap gambar dengan lingkungan setiap pixel dan menganggap hasilnya sebagai bilangan biner. Dengan menerapkan konsep LBP dan dikombinasikan dengan histogram, maka dapat dihasilkan suatu *feature descriptor* yang merepresentasikan *frame* wajah ke dalam bentuk data vektor sederhana.

F. Rekognisi Wajah

Proses selanjutnya adalah proses rekognisi wajah. Pada proses rekognisi wajah ini menerapkan algoritma LBPH, dimana wajah yang tertangkap kamera dan telah terdeteksi oleh classifier kemudian diubah menjadi grayscale secara otomatis oleh sistem secara real time dan kemudian nilai dari citra wajah tersebut dibandingkan dengan nilai hasil latih yang telah didapatkan sebelumnya oleh sistem admin. Pada proses metode ini akan menggunakan Euclidean Distrance untuk mencari nilai distance terdekat antara data LBPH pada citra input dan pada data LBPH yang telah dilatih dalam model, rumus untuk mencari Euclidean distance yaitu :

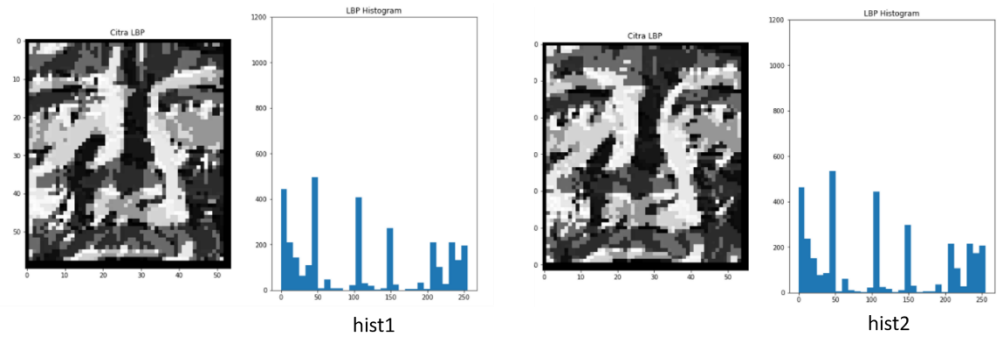
$$D = \sqrt{\sum_{i=1}^{n} (hist1_i - hist2_i)^2} \qquad (3.2)$$

Keterangan:

- D = Jarak antara histogram
- Hist1= nilai histogram yang telah dilatih
- Hist2= nilai histogram yang diuji
- N = dimensi input

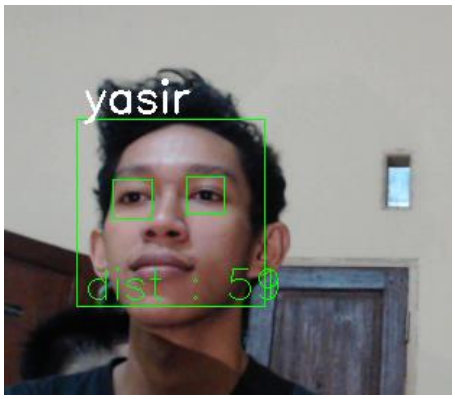
Dimana pada algoritma ini akan dilakukan pencarian ID histogram pada model yang terdekat pada nilai histogram pada data yang diproses. Dimana semakin kecil nilai dari distance maka semakin dekat kesamaan data histogram yang diinput dengan data histogram yang berada pada model. Dapat dilihat perbedaan nilai

histogram antara nilai histogram yang telah dilatih dalam model dan nilai histogram inputan citra pada gambar 3.21



Gambar 3.21 perbandingan antara histogram

Pada kedua histogram tersebut dimana hist1 merupakan histogram yang berasal pada model, sedangkan hist2 merupakan histogram yang berasal dari data inputan. Untuk kedua histogram tersebut akan dilakukan pencarian euclidean distance sehingga mendapatkan hasil seperti gambar 3.22.



Gambar 3.22 hasil proses euclidean distance

Pada gambar 3.22 dapat kita lihat hasil dari euclidean distance antara kedua histogram yang dimaksud adalah 59. Jika nilai distance yang didapatkan tidak melewati nilai threshold yang telah ditentukan, maka sistem akan mengenali orang yang sedang terbaca atau terdeteksi kamera tersebut. Begitu pula sebaliknya, jika

nilai distance dari prediksi algorima LBPH menjauhi atau lebih besar dari nilai distance yang telah ditentukan, maka sistem tidak akan mengenali orang yang terdeteksi oleh sistem.

G. Output

Output dari sistem presensi pegawai berbasi *face recognition* ini yaitu video yang tersusun dari *frame* hasil rekognisi wajah dengan tipe RGB yang telah diberikan bounding box serta dilabeli dengan nama pegawai yang terdektesi.

H. Menyimpan Data Prensensi ke Database

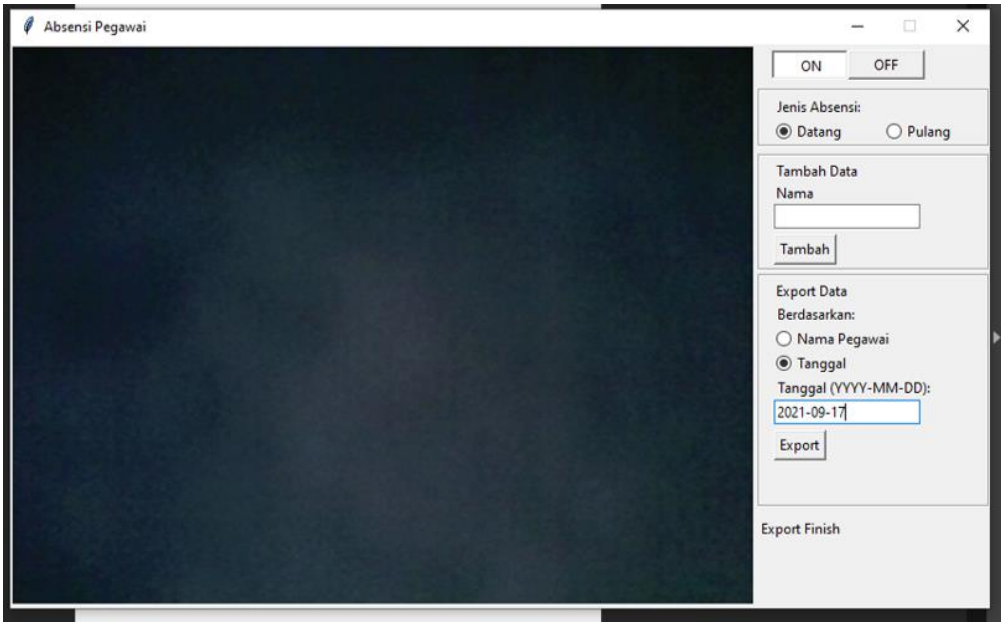
Sistem yang dibangun menggunakan MySQL untuk menyimpan data pegawai dan presensinya. Untuk menghubungkan sistem ke *database* digunakan *library* SQLAlchemy. SQLAlchemy merupakan SQL *toolkit* dan *Object Relational Mapper* (ORM) yang memungkinkan penggunaan semua fitur dan flexibilitas dari SQL.

Presensi pada sistem ini terbagi menjadi dua, yaitu presensi datang dan presensi pulang. Presensi datang dilakukan oleh pegawai pada jam masuk kantor, sedangkan presensi pulang dilakukan pada jam pulang kantor. Ketika seorang pegawai ingin melakukan presensi, maka dia harus menampakkan wajahnya di depan kamera. Pegawai tersebut akan dianggap hadir apabila wajahnya berhasil dideteksi dan dikenali oleh sistem. Setelah dianggap hadir, nama pegawai, tanggal dan waktu dilakukannya presensi akan dikirim ke *database*.

I. Graphical User Interface

Setelah sistem ini bekerja sesuai yang diharapkan, selanjutnya dilakukan proses pembuatan *Graphical User Interface* (GUI). Sistem ini menggunakan

library tkinter dari python. Pada *interface* dari sistem ini, terdapat beberapa komponen-komponen seperti *video stream* untuk menampilkan *input* dari *webcam*, tombol *ON/OFF* untuk mengaktifkan atau menonaktifkan presensi, *radio button* untuk memilih jenis presensi datang atau pulang, *form* tambah data untuk menambah data pegawai baru, *form export* data untuk mengunduh laporan presensi pegawai berdasarkan nama pegawai atau tanggal, dan *text log* yang menampilkan status dari aplikasi yang bisa di lihat di gambar 3.23.



Gambar 3.23 Ilustrasi Graphical User Interface

J. Pembuatan Laporan Presensi

Proses pembuatan laporan presensi dilakukan dengan menggunakan library Reportlab. Laporan yang dihasilkan berupa file dengan format pdf. Laporan presensi dapat di*export* berdasarkan nama pegawai atau tanggal tertentu. Laporan ini terdiri dari 6 kolom yaitu nomor, tanggal, nama pegawai, jam datang, jam pulang, dan status.

3.6 Analisis Kinerja Sistem

Untuk mengetahui kinerja sistem maka dilakukan pendataan hasil prediksi. Pada kasus klasifikasi kinerja sistem dapat diukur dengan menggunakan confusion matrix. Pada dasarnya, confusion matrix mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan doleh sistem dengan hasil klasifikasi yang sebenarnya. Pada analisis kinerja sistem menggunakan confusion matrix, terdapat empat istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut adalah True Positive (TP), True Negative (TN), False Positive (FP) dan False Negative (FN). Nilai True Negative (TN) merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan False Positive (FP) merupakan data negatif namun terdeteksi sebagai data positif. Sementara itu, True Positive (TP) merupakan data positif yang terdeteksi benar. False Negative (FN) merupakan kebalikan dari True Positive, sehingga data posifit, namun terdeteksi sebagai data negatif. Berdasarkan nilai True Negative (TN), False Positive (FP), False Negative (FN), dan True Positive (TP) dapat diperoleh nilai akurasi dimana data dilihat pada persamaan 3.2.

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$$

BAB IV

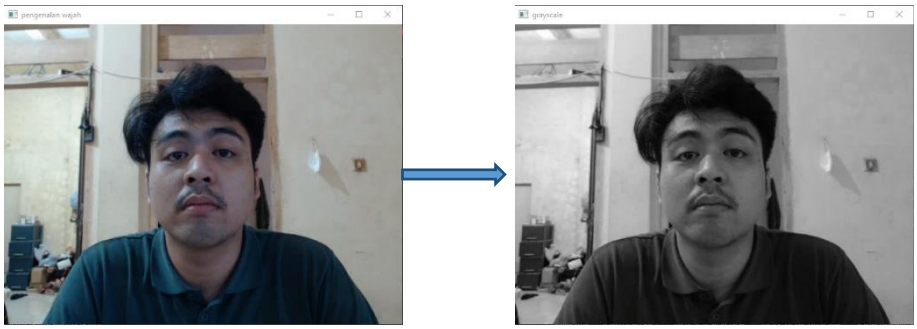
HASIL DAN PEMBAHASAN

4.1 Analisis Kinerja Sistem

4.1.1 Hasil Operasi Data Citra

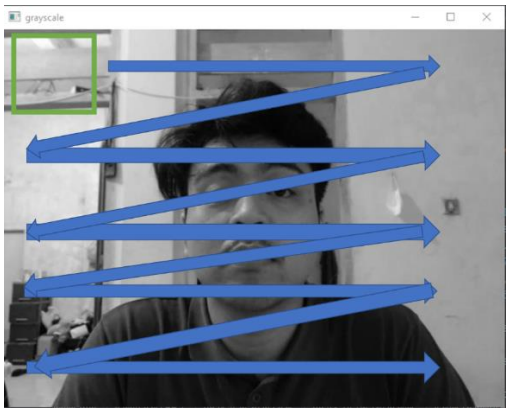
1. Hasil Deteksi Wajah pada *Frame*

Tahap mendeteksi wajah ini menggunakan metode Viola Jones. Metode ini melakukan pendeteksian wajah di rana gambar grayscale maka dari itu inputan citra di ubah menjadi grayscale terlebih dahulu.

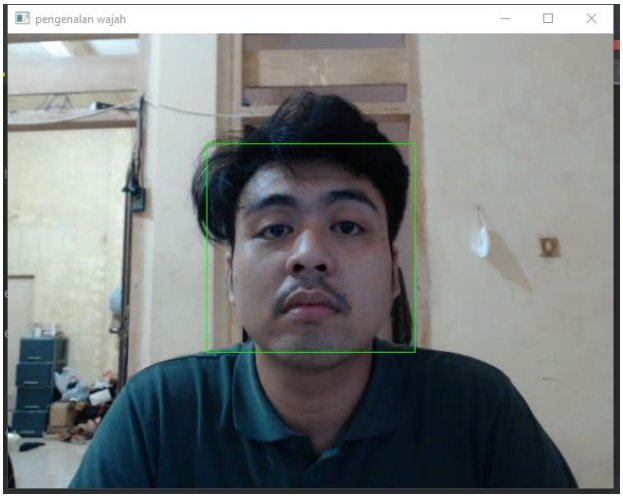


Gambar 4.1 merubah inputan citra menjadi grayscale

Tahap selanjutnya akan dilakukan proses sliding window untuk menemukan wajah pada input citra.



Gambar 4.2 ilustrasi proses sliding window

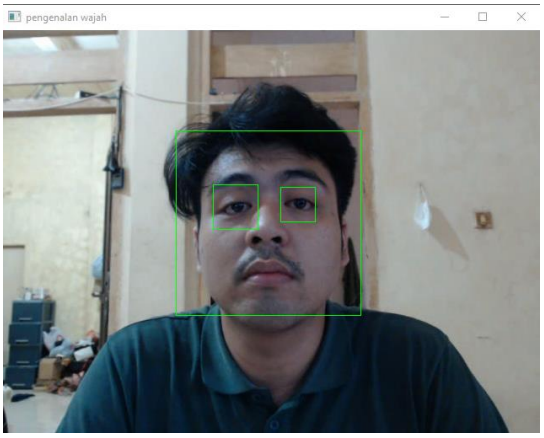


Gambar 4.3 Hasil deteksi wajah pada citra input

Jika pada citra input terdapat wajah, maka akan dibentuk ROI yang bertujuan untuk memperkecil area sliding window pada deteksi mata. Tetapi jika pada citra input tersebut tidak terdapat wajah, maka Sistem akan lanjut ke citra input berikutnya.

2. Hasil Deteksi Mata pada ROI

Tahap selanjutnya adalah deteksi mata untuk rotasi wajah. Setelah didapatkan ROI wajah dilakukan pendeteksian mata menggunakan metode Viola Jones.

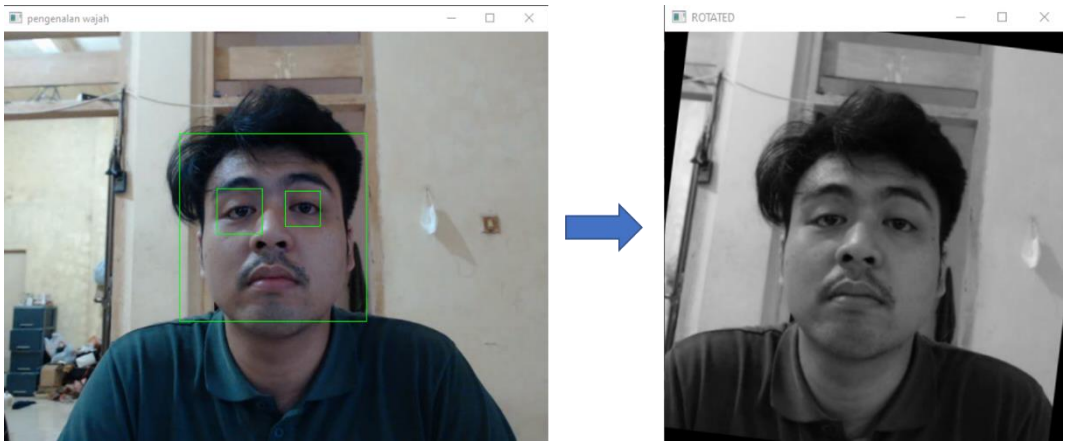


Gambar 4.4 hasil deteksi mata

Jika mata yang terdeksi ada dua maka sistem akan lanjut ke tahap berikutnya yaitu melakukan rotasi gambar. Jika tidak terdeteksi dua mata, maka sistem akan kembali untuk mencari wajah.

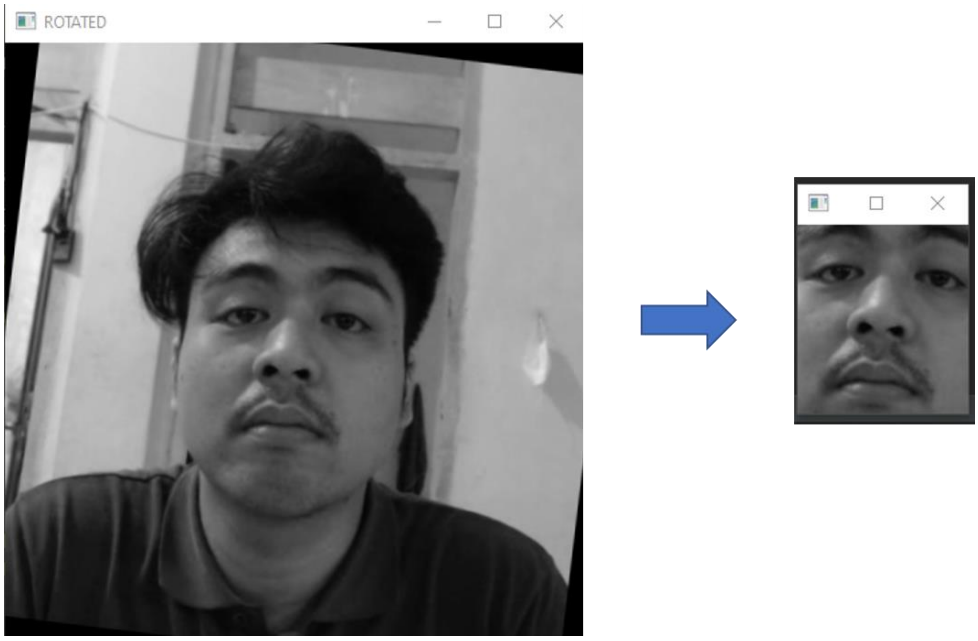
3. Rotasi Gambar

Rotasi gambar dilakukan dengan penghitungan titik tengah yang berada di antara kedua mata yang telah di deteksi pada proses sebelumnya. Titik tersebut akan digunakan pada perhitungan matrix rotasi dari frame sehingga kedua mata akan tampak sejajar secara horizontal.

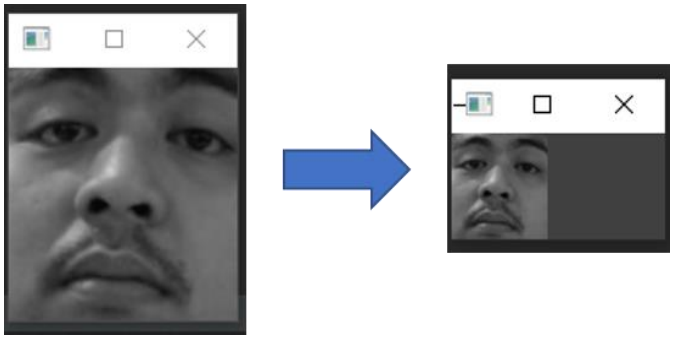


Gambar 4.5 ilustrasi proses rotasi wajah

Setelah dilakukan rotasi wajah dilakukan crop dan resize terhadap hasil rotasi untuk mendapatkan citra wajah yang akan digunakan untuk rekognisi wajah.



Gambar 4.6 hasil crop pada wajah terotasi

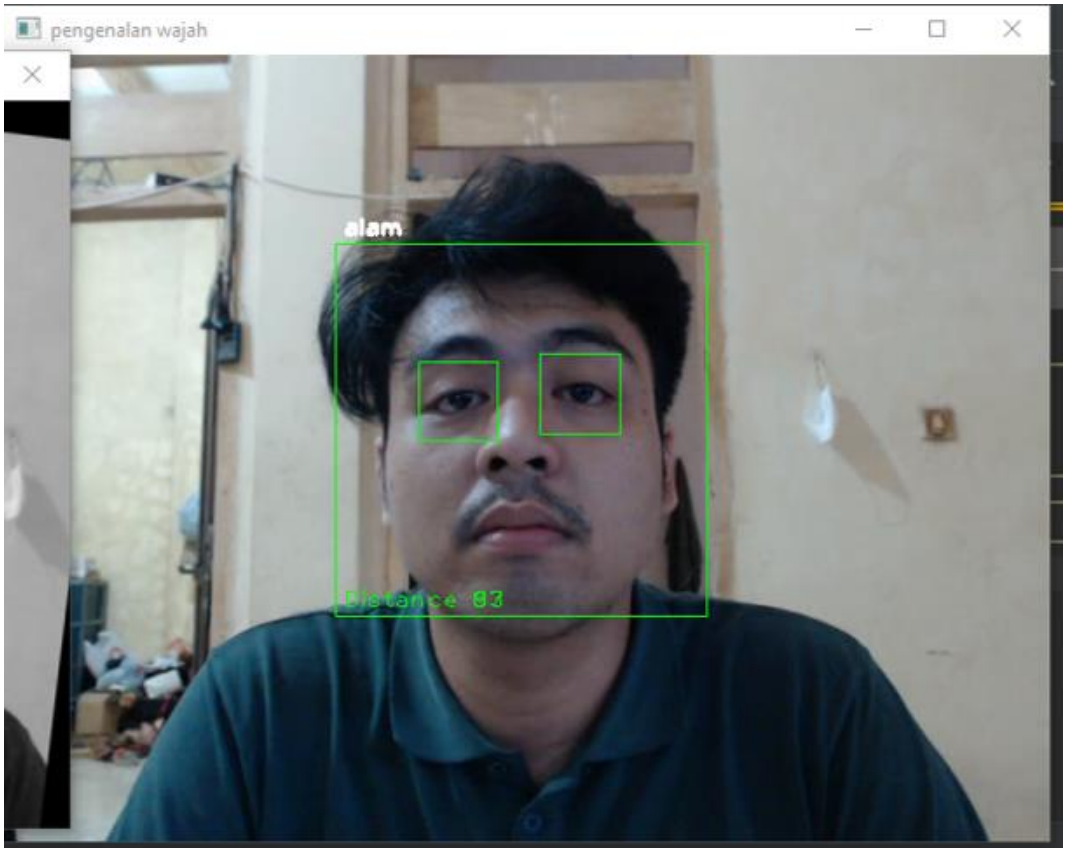


Gambar 4.7 hasil resize pada wajah crop

4. Rekognisi wajah

Tahap rekognisi merupaka tahap untuk mengenali wajah pegawai dengan cara menggunakan metode LBPH, dimana citra wajah yang telah dicrop akan diubah menjadi gambar LBP kemudian gambar tersebut di ekstraksi nilai Histogramnya. Nilai Histogram tersebut digunakan untuk mentukan jarak Euclidean untuk dibandingkan dengan data latih untuk mencari ID dengan nilai

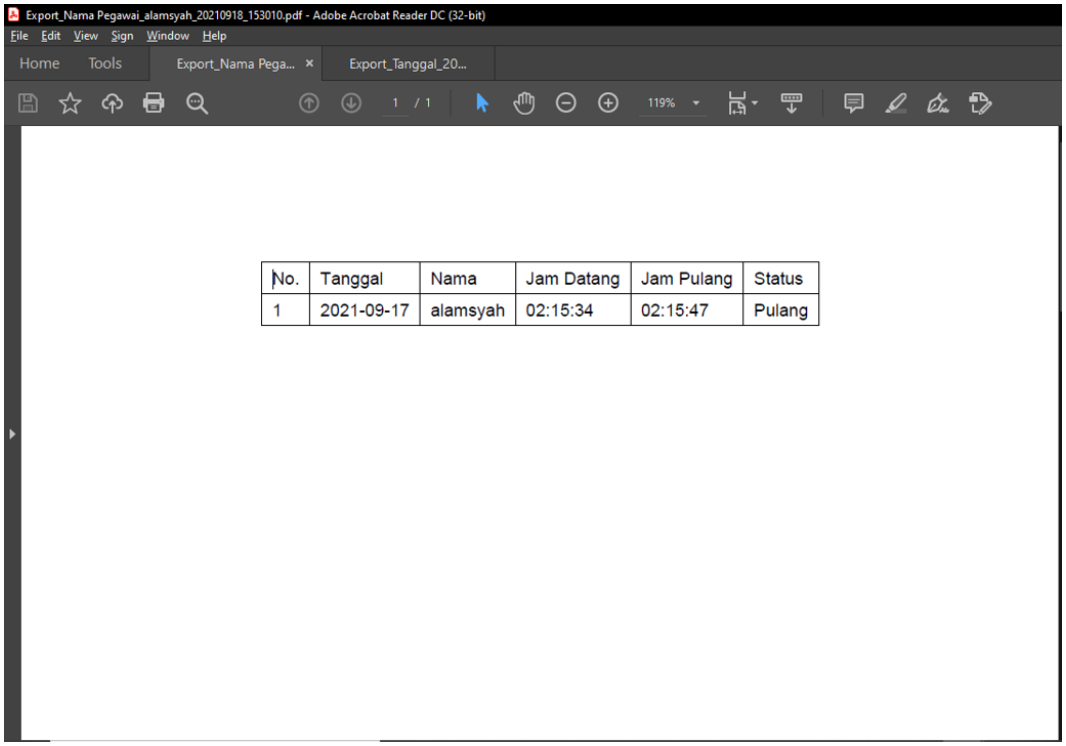
Euclidean terdekat.



Gambar 4.8 hasil rekognisi wajah

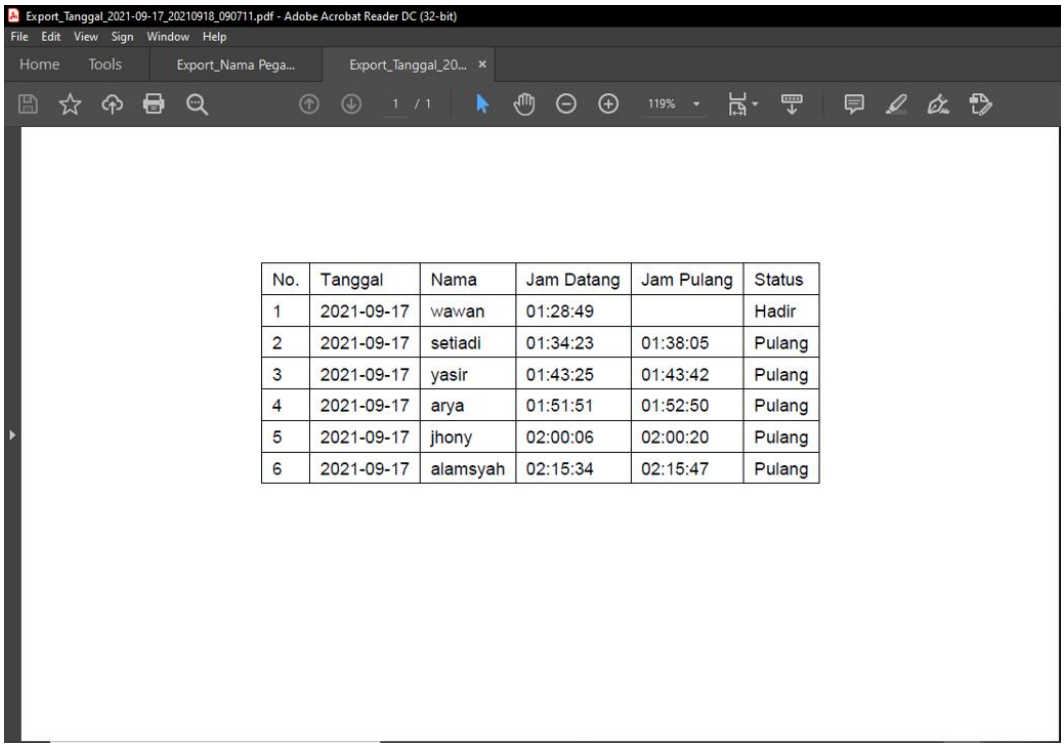
5. Hasil Pembuatan Laporan Presensi

Laporan presensi yang dihasilkan dari proses export data merupakan file dengan format pdf yang berisi data presensi pegawai. Data presensi pegawai dapat diexport berdasarkan nama pegawai ataupun tanggal. Pada file laporan tersebut, data presensi pegawai akan direpresentasikan dalam bentuk tabel yang terdiri dari kolom nomor, tanggal, nama pegawai, jam datang, jam pulang, dan status. File yang dihasilkan memiliki nama yang unik dengan *template* "Export_(Jenis Export)_(Kata Kunci)_(Tanggal)_(Waktu).pdf".



No.	Tanggal	Nama	Jam Datang	Jam Pulang	Status
1	2021-09-17	alamsyah	02:15:34	02:15:47	Pulang

Gambar 4.9 contoh hasil laporan berdasarkan individu



No.	Tanggal	Nama	Jam Datang	Jam Pulang	Status
1	2021-09-17	wawan	01:28:49		Hadir
2	2021-09-17	setiadi	01:34:23	01:38:05	Pulang
3	2021-09-17	yasir	01:43:25	01:43:42	Pulang
4	2021-09-17	arya	01:51:51	01:52:50	Pulang
5	2021-09-17	jhony	02:00:06	02:00:20	Pulang
6	2021-09-17	alamsyah	02:15:34	02:15:47	Pulang

Gambar 4.10 contoh hasil laporan berdasarkan tanggal

4.1.2 Hasil Pengujian sistem

Pada pengujian pengenalan sistem ingin diketahui bagaimana sistem mengenali wajah dan tingkat kesalah dari pengenalannya juga seberapa cepat proses pengenalan wajah yang dilakukan oleh sistem. Pada Tabel 4.1 diperlihatkan ada enam data didalam database yang disimbolkan dengan label berbentuk angka 0-5, masing-masing data memiliki data citra pegawai didalamnya. Setiap data pegawai diuji sebanyak lima kali dan dihitung berapa kali wajah dikenali dalam lika kali percobaan tersebut. Percobaan dilakukan dengan menguji kamera membaca gambar wajah dari anggota yang di representasikan oleh masing-masing label.

Tabel 4.1 Tabel pengujian wajah dikenali oleh sistem

Data dalam database	Jumlah Dikenali (TP)	Tidak Dikenali (FN)
Data 0	5	0
Data 1	5	0
Data 2	5	0
Data 3	5	0
Data 4	5	0
Data 5	3	2

Pengujian juga dilakukan kepada sistem untuk data atau gambar wajah yang tidak ada didalam database. Percobaan dilakukan dengan sistem dengan wajah orang sebanyak tiga yang tidak terdaftar dalam database dengan masing-masing percobaan dilakukan sebanyak lima kali. Sebagaimana dapat dilihat pada table 4.2

Tabel 4.2 Tabel pengujian wajah yang tidak terdapat didalam database

Data tidak terdaftar didalam database	Jumlah Dikenali (FP)	Tidak Dikenali (TN)
Data 0	0	5
Data 1	0	5
Data 2	0	5

$$\text{Total Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$$

$$= \frac{53}{55} \times 100\%$$

$$= 96{,}36\%$$

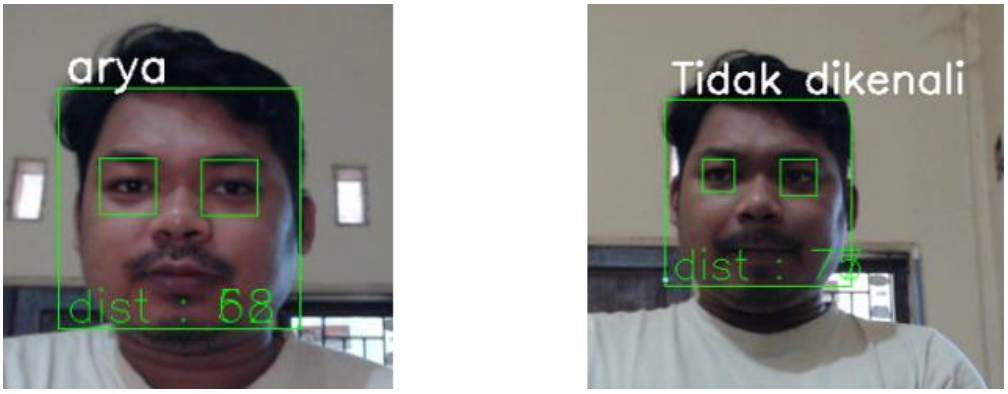
Pengujian juga dilakukan untuk seberapa cepat waktu yang dibutuhkan sistem untuk mengenali sebuah wajah dari pertama deteksi wajah hingga selesai proses rekognisi wajah. Data tersebut diperlihatkan pada table 4.3.

Tabel 4.3 Tabel waktu proses rekognisi wajah

Data dalam database	Waktu proses di setiap pengujian (detik)				
	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
Data 0	0,294	0,267	0,182	0,194	0,191
Data 1	0,208	0,177	0,262	0,24	0,23
Data 2	0,144	0,299	0,202	0,284	0,171
Data 3	0,190	0,131	0,136	0,133	0,107
Data 4	0,294	0,267	0,182	0,194	0,191
Data 5	0,233	Tidak terekognisi	0,197	Tidak terekognisi	0,148

Dari tabel 4.3 dapat dilihat bahwa sistem dengan menggunakan laju 30fps membutuhkan rata – rata waktu proses selama 0.19 detik.

Pada gambar 4.11 terjadinya gagal rekognisi disebabkan karena kurangnya pencahayaan pada wajah disaat akan dilakukan rekognisi wajah. Ini menyebabkan nilai distance menjadi naik sehingga menyebabkan wajah tidak dapat dikenali.



Gambar 4.11 contoh wajah gagal terekognisi

4.2 Pembahasan

Pada tabel 4.1 dan tabel 4.2 dapat dilihat bahwa tingkat akurasi dari pengenalan wajah yaitu 96,36%. Adapun hal-hal yang memungkinkan mempengaruhi pengenalan yaitu tingkat intensitas cahaya pada wajah atau ruangan dan terdapat aksesoris pada wajah seperti yang dapat dilihat pada gambar 4.12.

Gambar 4.12 perbandingan rekognisi wajah menggunakan aksesoris.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan maka dapat ditarik kesimpulan sebagai berikut.

1. Pengenalan wajah karyawan menggunakan computer vision dapat dilkaukan dengan mneerapkan metode Viola-Jones, dan Local Binary Pattern Histogram.
2. Mendata karyawan yang telah hadir dan yang telah istirahat dapat dilakukan dengan membangun antarmuka aplikasi yang menggunakan metode pengenalan wajah, dimana pada aplikasi tersebut user dapat mengganti tipe input presensi berupa pegawai datang atau pegawai istirahat. Hasil presensi pegawai disimpan di database lokal yang kemudian digunakan untuk membuat laporan presensi pegawai per-orang ataupun per-tanggal. Hasil pengujian sistem rekognisi wajah pegawai menunjukkan dengan jumlah data uji sebanyak 6 orang, dimana setiap orang memiliki 30 data latih, sistem berhasil melakukan presensi pegawai dengan tingkat akurasi sebesar 96,36%, dengan rata-rata kecepatan deteksi sebesar 0,19 detik.

5.2 Saran

Berdasarkan penelitian yang dilakukan maka penulis memberikan saran sebagai berikut.

1. Sistem tidak dapat mendeteksi wajah jika kondisi cahaya kurang baik, sehingga diharapkan pada penelitian kedepannya sistem dapat mendeteksi wajah walau dalam kondisi cahaya kurang baik
2. Sistem mendeteksi dan mengenali wajah dalam bentuk gambar yang dicetak, sehingga diharapkan pada penelitian kedepannya sistem dapat membedakan atau tidak dapat mandates wajah jika dalam berbentuk gambar yang dicetak.
3. Sistem menggunakan database lokal untuk penyimpanan presensi, sehingga diharapkan pada penelitian kedepannya dapat dibuat sistem presensi yang berbasis online

DAFTAR PUSTAKA

[3] Kalansuriya, T. R., & Dharmaratne, A. T. (2013). Facial Image Classification Based on Age and Gender, 44–50.

[4] Wulansari, D., Djamal, E. C., & Ilyas, R. (2017). Identifikasi Gender Berdasarkan Citra Wajah Menggunakan Deteksi Tepi dan Backpropagation, 10–14.

[5] Kalansuriya, T. R., & Dharmaratne, A. T. (2015). Neural Network based Age and Gender Classification for Facial Images, (April). https://doi.org/10.4038/icter.v7i2.7154

[6] Shinichi Terada. (2018). Firms strop goods as facial hair finds favor, 1–4.

[7] Sachdeva, S. (2018). HIRSUTISM : EVALUATION AND TREATMENT, 55(1), 3–7. https://doi.org/10.4103/0019-5154.60342

[8] Szeliski, R. 2010. Computer Vision: Algorithms and Applications. Springer

[9] Zhao, W. (2006). Face processing: advanced modeling and methods. Journal of Electronic ….

[10] Putra, Darma. 2010. Pengolahan Citra Digital. Yogyakarta: Andi.

[11] Mukhopadhyay, Jayanta. 2011. Image and Video Processing in the Compressed Domain. Chapman and Hall/CRC.

[12] Shulur dan Amalga, 2015. Perancangan Aplikasi Deteksi Wajah Menggunakan Algoritma Viola-Jones, Bandung: Universitas Pasundan.

[13] Li, Stan Z & Jain, Anil K. 2011. Handbook of Face Recognition. Springer

[14] Lopez, L.S. 2010. Local Binary Patterns applied to Face Detection and Recognition. Universitat Politecnica De Catalunya

[15] Prasetyo, E. 2014. Data Mining Mengubah Data Menjadi Informasi Menggunakan Matlab. Penerbit Andi. Yogyakarta.

[16] Prado, Kelvin Salton do. 2017. Face Recognition: Understanding LBPH Algorithm. https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b. diakses pada 11 Juli 2021

[17] python.org

[18] opencv.org

LAMPIRAN

- **Source code program**

a) Source code fungsi deteksi wajah untuk data latih

```
from threading import Thread
import cv2
import config
from ml.face_function import face_rotate
from ml.train_dataset import generate_face_xml

total_sample = config.TOTAL_SAMPLE


class GenerateDataset(Thread):
    def __init__(self, user_id, video_comp):
        super().__init__()

        self.sample_number = 0
        self.user_id = user_id
        self.video_comp = video_comp

    def run(self):
        detector_path = 'haarcascade_frontalface_default.xml'
        detector = cv2.CascadeClassifier(detector_path)
        sf = 1.1
        mn = 8

        while True:
            _, image = self.video_comp.vid.read()
            frame_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(frame_gray, scaleFactor=sf, minNeighbors=mn,
                                              minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)
```

```
            for (x, y, w, h) in faces:
                eye_gray = frame_gray[y - int(h / 2): y + int(h * 1.5), x - int(x / 2): x + int(w * 1.5)]
                img = (face_rotate(eye_gray))
                if img is not None:
                    frame = img
                    cv2.imwrite(
                        "training_data{0}/user.{1}.{2}.jpg".format(total_sample, str(self.user_id),
                                                                   str(self.sample_number)),
                        frame)
                    print("Ekstrak frame dengan wajah : {0} dari {1}".format(str(self.sample_number), total_sample))
                    self.sample_number = self.sample_number + 1
            if self.sample_number > total_sample:
                break
        print('--------------------')
        print('Ekstraksi selesai')
        generate_face_xml()
```

b) Source code fungsi rotasi wajah

```python
import cv2
import math


def face_rotate(image):
    haar = 'haarcascade_frontalface_default.xml'
    haar_eye = 'haarcascade_eye.xml'
    face_detector = cv2.CascadeClassifier(haar)
    eye_detector = cv2.CascadeClassifier(haar_eye)

    Theta = 0
    rows, cols = image.shape
    eyes = eye_detector.detectMultiScale(image)
    for (sx, sy, sw, sh) in eyes:
        # input berupa dua mata
        if eyes.shape[0] == 2:
            # menentukan mata kiri dan kanan
            if eyes[1][0] > eyes[0][0]:
                # beda tinggi di antara mata
                DY = ((eyes[1][1] + eyes[1][3] / 2) - (eyes[0][1] + eyes[0][3] / 2))
                # beda lebar antara mata
                DX = ((eyes[1][0] + eyes[1][2] / 2) - eyes[0][0] + (eyes[0][2] / 2))
            else:
                # beda tinggi antara mata
                DY = (-(eyes[1][1] + eyes[1][3] / 2) + (eyes[0][1] + eyes[0][3] / 2))
                # beda wajah antara mata
                DX = (-(eyes[1][0] + eyes[1][2] / 2) + eyes[0][0] + (eyes[0][2] / 2))
```

```python

            # memastikan rotasi jika diperlukan
            if (DX != 0.0) and (DY != 0.0):
                # perhitungan sudut euclidian
                Theta = math.degrees(math.atan(round(float(DY) / float(DX), 2)))
                print("Theta  ", str(Theta))

            M = cv2.getRotationMatrix2D((cols / 2, rows / 2), Theta, 1)  # menentukan matrix rotasi
            image = cv2.warpAffine(image, M, (cols, rows))
            # cv2.imshow('ROTATED', Image)                                        # rotasi wajah

            Face2 = face_detector.detectMultiScale(image, 1.1, 8)  # deteksi wajah
            for (FaceX, FaceY, FaceWidth, FaceHeight) in Face2:
                CroppedFace = image[FaceY + int(FaceHeight / 4.7): FaceY + int(FaceHeight - int(FaceHeight / 6.5)),
                              FaceX + int(FaceWidth / 4.7):FaceX + int(FaceWidth - int(FaceWidth / 4.7))]
                return CroppedFace
```

c) Source code fungsi latih model rekognisi wajah

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


def get_image_label(path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faceSamples = []
    faceIDs = []
    for imagePath in imagePaths:
        PILImg = Image.open(imagePath).convert('L')
        PILImg = PILImg.resize((54, 59))
        imgNum = np.array(PILImg, 'uint8')
        faceID = int(os.path.split(imagePath)[-1].split('.')[1])
        faceSamples.append(imgNum)
        faceIDs.append(faceID)

    return np.array(faceIDs), faceSamples



def generate_face_xml():
    training_path = "training_data{0}".format(config.TOTAL_SAMPLE)
    face_Recognizer = cv2.face.LBPHFaceRecognizer_create(1, 6, 8, 8)
    face_IDs, FaceSamples = get_image_label(training_path)

    print('TRAINING......')
    face_Recognizer.train(FaceSamples, face_IDs)
    face_Recognizer.save("training{0}.xml".format(config.TOTAL_SAMPLE))
    print('XML FILE SAVED....')

```

d) Source code GUI

```python
from enum import Enum
from datetime import datetime
import math

from statistics import mode
from tkinter import *
from tkinter import ttk, messagebox
from PIL import Image, ImageTk
from reportlab.lib import colors
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
import csv
import cv2
import time

from db.crud import crud_user, crud_absensi
from db.models.absensi import AbsensiType
from db.session import get_db
from ml.generate_dataset import GenerateDataset
import config

total_sample = config.TOTAL_SAMPLE
haar = 'haarcascade_frontalface_default.xml'
haar_eye = 'haarcascade_eye.xml'
face_detector = cv2.CascadeClassifier(haar)
eye_detector = cv2.CascadeClassifier(haar_eye)
font = cv2.FONT_HERSHEY_SIMPLEX
```

```python
def get_list_absensi_user(user_id):
    with get_db() as db:
        return crud_absensi.get_list_absensi_by_user_id(db, user_id)


def get_list_absensi_date(date):
    with get_db() as db:
        return crud_absensi.get_list_absensi_by_date(db, date)


def get_user_absensi_today(user_id):
    with get_db() as db:
        return crud_absensi.get_absensi_today_by_user_id(db, user_id)


def add_absensi(absensi_type, user_id, curent_datetime):
    absensi_type = AbsensiType(absensi_type)
    with get_db() as db:
        dtstring_date = curent_datetime.strftime("%Y-%m-%d")
        dtstring_hour = curent_datetime.strftime("%H:%M:%S")
        crud_absensi.create_absensi(db, user_id=user_id, type=absensi_type, date=dtstring_date, time=dtstring_hour)


def get_all_users():
    with get_db() as db:
        return crud_user.get_list_users(db)
```

```python
def create_user(name):
    with get_db() as db:
        return crud_user.create_user(db, name=name)


def get_option_user_dicts():
    return {user.name: user.id for user in get_all_users()}


def get_list_user_names():
    return ["", *[user.name for user in get_all_users()]]


class ExportType(str, Enum):
    USER = "Nama Pegawai"
    DATE = "Tanggal"


```

```python
class App(Tk):
    def __init__(self, video_source):
        super().__init__()

        self.bind('<Escape>', lambda e: self.quit())
        self.title("Absensi Pegawai")
        self.resizable(0, 0)

        self.hadir_analisis = []
        self.hadir_insert = []
        self.hadir_name = "Tidak Dikenali"
        self.count = 0

        self.face_recognizer = cv2.face.LBPHFaceRecognizer_create(1, 6, 8, 8)
        self.face_recognizer.read('training{0}.xml'.format(total_sample))

        self.selected_user = StringVar()
        self.selected_user.set("-- Pilih --")

        self.list_user_names = get_list_user_names()
        self.option_users_dict = get_option_user_dicts()

        # open video source (by default this will try to open the computer webcam)
        self.video_comp = VideoCaptureComp(video_source)

        # Create a canvas that can fit the above video source size
        self.canvas = Canvas(self, width=self.video_comp.width, height=self.video_comp.height)
        self.canvas.grid(column=0, row=0, rowspan=15)

```

```python
        # SWITCH
        self.face_recognition_switch = BooleanVar(value=True)
        self.switch_frame = ttk.Frame(self, padding=(12, 3, 12, 3))
        self.switch_frame.grid(column=1, row=0, sticky=W)
        self.on_button = Radiobutton(self.switch_frame,
                                     text="ON",
                                     variable=self.face_recognition_switch,
                                     indicatoron=False,
                                     value=True,
                                     width=8,
                                     command=self.on_switch_change)
        self.off_button = Radiobutton(self.switch_frame,
                                      text="OFF",
                                      variable=self.face_recognition_switch,
                                      indicatoron=False,
                                      value=False,
                                      width=8,
                                      command=self.on_switch_change)
        self.on_button.grid(column=0, row=0)
        self.off_button.grid(column=1, row=0)

```

```
        # ABSENSI
        self.absensi_type = StringVar()
        self.absensi_type.set(AbsensiType.CHECK_IN.value)
        self.frame_absensi_type = ttk.Frame(self,
                                            borderwidth=2,
                                            relief="groove",
                                            width=200,
                                            height=50,
                                            padding=(12, 3, 12, 3))
        self.frame_absensi_type.configure(height=self.frame_absensi_type["height"],
                                          width=self.frame_absensi_type["width"])
        self.frame_absensi_type.grid_propagate(0)
        self.frame_absensi_type.grid(column=1, row=1, sticky=W)
        self.label_type_absensi = Label(self.frame_absensi_type,
                                        text="Jenis Absensi:",
                                        anchor='w')
        self.rb_datang = ttk.Radiobutton(self.frame_absensi_type,
                                         text=AbsensiType.CHECK_IN.value,
                                         variable=self.absensi_type,
                                         value=AbsensiType.CHECK_IN.value,
                                         command=self.on_absensi_type_change)
```

```
        self.rb_pulang = ttk.Radiobutton(self.frame_absensi_type,
                                         text=AbsensiType.CHECK_OUT.value,
                                         variable=self.absensi_type,
                                         value=AbsensiType.CHECK_OUT.value,
                                         command=self.on_absensi_type_change)
        self.label_type_absensi.grid(column=1, row=0, sticky=W)
        self.rb_datang.grid(column=1, row=1, sticky=W)
        self.rb_pulang.grid(column=2, row=1, sticky=W)
        self.frame_absensi_type.columnconfigure(1, weight=1)
        self.frame_absensi_type.columnconfigure(2, weight=1)

```

```
        # ADD NEW USER
        self.user_name = StringVar()
        self.frame_new_user = ttk.Frame(self,
                                        borderwidth=2,
                                        relief="groove",
                                        width=200,
                                        height=100,
                                        padding=(12, 3, 12, 3))
        self.frame_new_user.configure(height=self.frame_new_user["height"], width=self.frame_new_user["width"])
        self.frame_new_user.grid_propagate(0)
        self.frame_new_user.grid(column=1, row=2, sticky=W)
        self.label_add_data = ttk.Label(self.frame_new_user, text="Tambah Data")
        self.label_name = ttk.Label(self.frame_new_user, text="Nama")
        self.input_name = ttk.Entry(self.frame_new_user, textvariable=self.user_name)
        self.button_add_data = Button(self.frame_new_user,
                                      text='Tambah',
                                      command=(lambda value=self.input_name: self.add_new_user(user_name=value.get())))
        self.label_add_data.grid(column=1, row=0, sticky=W)
        self.label_name.grid(column=1, row=1, sticky=W)
        self.input_name.grid(column=1, row=2)
        self.button_add_data.grid(column=1, row=3, pady=5, sticky=W)

```

```
        # EXPORT
        self.export_type = StringVar()
        self.selected_date = StringVar()
        self.export_type.set(ExportType.USER.value)
        self.frame_export = ttk.Frame(self,
                                      borderwidth=2,
                                      relief="groove",
                                      width=200,
                                      height=200,
                                      padding=(12, 3, 12, 3))
        self.frame_export.configure(height=self.frame_export["height"], width=self.frame_export["width"])
        self.frame_export.grid_propagate(0)
        self.frame_export.grid(column=1, row=3, sticky=NW)
        self.label_export_data = ttk.Label(self.frame_export, text="Export Data")
        self.label_type_export = Label(self.frame_export,
                                       text="Berdasarkan:",
                                       anchor='w')
        self.rb_user = ttk.Radiobutton(self.frame_export,
                                       text=ExportType.USER.value,
                                       variable=self.export_type,
                                       value=ExportType.USER.value,
                                       command=self.on_export_type_change)
```

```
        self.rb_date = ttk.Radiobutton(self.frame_export,
                                       text=ExportType.DATE.value,
                                       variable=self.export_type,
                                       value=ExportType.DATE.value,
                                       command=self.on_export_type_change)
        self.label_select_user = Label(self.frame_export, text="Pilih user:")
        self.option_users = OptionMenu(self.frame_export, self.selected_user, *self.option_users_dict.keys())
        self.label_date = Label(self.frame_export, text="Tanggal (YYYY-MM-DD):")
        self.input_date = ttk.Entry(self.frame_export, textvariable=self.selected_date)
        self.button_export = Button(self.frame_export, text='Export',
                                    command=(lambda value=self.export_type: self.export_absensi(value.get())))
        self.label_export_data.grid(column=1, row=0, sticky=W)
        self.label_type_export.grid(column=1, row=1, sticky=W)
        self.rb_user.grid(column=1, row=2, sticky=W)
        self.rb_date.grid(column=1, row=3, sticky=W)
        self.button_export.grid(column=1, row=6, pady=5, sticky=W)
```

```
        # INFO MESSAGE
        self.info_message = StringVar()
        self.label_info = Label(self, textvariable=self.info_message)
        self.label_info.grid(column=1, row=4, sticky=W)

        self.on_absensi_type_change()
        self.on_export_type_change()

        # After it is called once, the update method will be automatically called every delay milliseconds
        self.delay = 50
        self.update_frame()

    def refresh_face_recognizer(self):
        self.face_recognizer = cv2.face.LBPHFaceRecognizer_create(1, 6, 8, 8)
        self.face_recognizer.read('training{0}.xml'.format(total_sample))

    def update_info_message(self, message):
        self.info_message.set(message)

    def on_absensi_type_change(self):
        selected_absensi_type = self.absensi_type.get()
        message = "Absensi {0} Aktif".format(selected_absensi_type)
        self.update_info_message(message)
        print(message)

    def on_change_selected_user(self, selected_user):
        self.selected_user.set(selected_user)
```

```
244     def on_change_selected_user(self, selected_user):
245         self.selected_user.set(selected_user)
246 
247     def on_export_type_change(self):
248         selected_export_type = self.export_type.get()
249         if selected_export_type == ExportType.USER.value:
250             self.label_date.grid_remove()
251             self.input_date.grid_remove()
252             self.label_select_user.grid(column=1, row=4, sticky=W)
253             self.option_users.grid(column=1, row=5, sticky=W)
254         else:
255             self.label_select_user.grid_remove()
256             self.option_users.grid_remove()
257             self.label_date.grid(column=1, row=4, sticky=W)
258             self.input_date.grid(column=1, row=5, sticky=W)
259 
260     def on_switch_change(self):
261         face_recognition_active = self.face_recognition_switch.get()
262         if face_recognition_active:
263             message = "Absensi Diaktifkan"
264             self.refresh_face_recognizer()
265         else:
266             message = "Absensi Dinonaktifkan"
267         print(message)
268         self.update_info_message(message)
269 
```

e) Source code rekognisi wajah

```
269 
270     def update_frame(self):
271         # Get a frame from the video source
272         ret, frame = self.video_comp.get_frame()
273 
274         face_recognition_active = self.face_recognition_switch.get()
275         if face_recognition_active:
276             frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
277             faces = face_detector.detectMultiScale(frame_gray, scaleFactor=1.1, minNeighbors=8, minSize=(30, 30))
278             if len(faces) == 1:
279                 self.count = 0
280                 time_start = time.perf_counter()
281                 for (x, y, w, h) in faces:
282                     face = frame[y - int(h / 2): y + int(h * 1.5), x - int(x / 2): x + int(w * 1.5)]
283                     eye_gray = frame_gray[y - int(h / 2): y + int(h * 1.5), x - int(x / 2): x + int(w * 1.5)]
284                     eye_image = eye_gray
285 
286                     # Bounding box wajah
287                     cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 1)
288 
289                     rows, cols = eye_image.shape
290                     eyes = eye_detector.detectMultiScale(eye_image, 1.3, 5)
291                     for (sx, sy, sw, sh) in eyes:
292                         # Bounding box mata
293                         cv2.rectangle(face, (sx, sy), (sx + sw, sy + sh), (0, 255, 0), 1)
294                         # input berupa dua mata
295                         if eyes.shape[0] == 2:
296                             # menentukan mata kiri dan kanan
297                             if eyes[1][0] > eyes[0][0]:
```

```
297                 if eyes[1][0] > eyes[0][0]:
298                     # beda tinggi di antara mata
299                     DY = ((eyes[1][1] + eyes[1][3] / 2) - (eyes[0][1] + eyes[0][3] / 2))
300                     # beda lebar antara mata
301                     DX = ((eyes[1][0] + eyes[1][2] / 2) - eyes[0][0] + (eyes[0][2] / 2))
302                 else:
303                     # beda tinggi antara mata
304                     DY = (-(eyes[1][1] + eyes[1][3] / 2) + (eyes[0][1] + eyes[0][3] / 2))
305                     # beda wajah antara mata
306                     DX = (-(eyes[1][0] + eyes[1][2] / 2) + eyes[0][0] + (eyes[0][2] / 2))
307
308                 # memastikan rotasi jika diperlukan
309                 if (DX != 0.0) and (DY != 0.0):
310                     # perhitungan sudut euclidian
311                     Theta = math.degrees(math.atan(round(float(DY) / float(DX), 2)))
312                     print("Theta  ", str(Theta))
313
314                     M = cv2.getRotationMatrix2D((cols / 2, rows / 2), Theta, 1)  # menentukan matrix rotasi
315                     eye_image = cv2.warpAffine(eye_image, M, (cols, rows))
```

```
317                     Face2 = face_detector.detectMultiScale(eye_image, 1.1, 8)  # deteksi wajah untuk isolasi wajah
318                     for (FaceX, FaceY, FaceWidth, FaceHeight) in Face2:
319                         CroppedFace = eye_image[FaceY + int(FaceHeight / 4.7): FaceY + int(
320                             FaceHeight - int(FaceHeight / 6.5)),
321                                       FaceX + int(FaceWidth / 4.7):FaceX + int(
322                                           FaceWidth - int(FaceWidth / 4.7))]
323                         CroppedFace = cv2.resize(CroppedFace, (54, 59))
324                         # Bounding box wajah
325                         # cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 1)
326
327                         id, distance = self.face_recognizer.predict(CroppedFace)
328                         if distance >= 70:
329
330                             NameID = 0
331                             distanceTxt = "dist : {0}".format(round(distance))
332                             print('distance :' + str(round(distance)))
333                             print('id :' + str(NameID))
334                             self.hadir_analisis.append(NameID)
335
336                         else:
337                             NameID = id
338                             distanceTxt = "dist : {0}".format(round(distance))
339                             print('distance :' + str(round(distance)))
340                             print('id :' + str(NameID))
341
```

```
                    if len(self.hadir_analisis) < 15:
                        self.hadir_analisis.append(NameID)
                        print('analisis data :' + str(self.hadir_analisis))
                    else:
                        detected_user_id = mode(self.hadir_analisis)
                        total_detected_user_id = self.hadir_analisis.count(detected_user_id)
                        total_detected_unknown = self.hadir_analisis.count(0)
                        is_accepted = total_detected_user_id + total_detected_unknown >= 10
                        if is_accepted:
                            print('terbanyak muncul :' + str(detected_user_id))

                            current_datetime = datetime.now()
                            dtstring_date = current_datetime.strftime("%d/%m/%Y")
                            dtstring_time = current_datetime.strftime("%H:%M:%S")
                            dtstring_hour_minute = current_datetime.strftime("%H:%M")

                            if detected_user_id == 0 or detected_user_id >= len(self.list_user_names):
                                self.hadir_name = 'Tidak dikenali'
                            else:
                                self.hadir_name = self.list_user_names[detected_user_id]
                                absensi_type = AbsensiType(self.absensi_type.get())
                                user_absensi = get_user_absensi_today(detected_user_id)
                                if (user_absensi is None and absensi_type == AbsensiType.CHECK_IN) \
                                        or (user_absensi and absensi_type == AbsensiType.CHECK_OUT):
                                    add_absensi(self.absensi_type.get(), detected_user_id,
                                                current_datetime)
                                    time_elapsed = (time.perf_counter() - time_start)
                                    print('waktu proses :' + str(time_elapsed))
                                    message = "{0} {1} pada pukul {2}".format(self.hadir_name,
                                                                              self.absensi_type.get().lower(),
                                                                              dtstring_hour_minute)
                                    self.update_info_message(message)

                                    self.hadir_insert.append(self.hadir_name)
                                    self.hadir_insert.append(dtstring_time)
                                    self.hadir_insert.append(dtstring_date)
                                    print(self.hadir_insert)

                                    with open('daftarhadirdatang.csv', 'a+', newline='') as csv_file:
                                        writer = csv.writer(csv_file, delimiter=',')
                                        writer.writerow(self.hadir_insert)

                                    '''oldcsv = pd.read_csv('daftarhadirdatang.csv')
                                    newcsv = oldcsv.drop_duplicates(subset = 'name')'''

                                    print(str(self.hadir_name) + ' melakukan absen pada :' + dtstring_time)
                                self.hadir_analisis.clear()
                                self.hadir_insert.clear()

                    cv2.putText(frame, str(self.hadir_name), (x + 5, y - 5), font, 1, (255, 255, 255), 2)
                    cv2.putText(frame, str(distanceTxt), (x + 5, y + h - 5), font, 1, (0, 255, 0), 1)
```

```
            if len(faces) == (0):
                self.count = self.count + 1
                if self.count>10:
                    self.count=0
                    self.hadir_name='tidak dikenali'
                    self.hadir_insert.clear()
                    self.hadir_analisis.clear()

        if ret:
            self.photo = ImageTk.PhotoImage(image=Image.fromarray(frame))
            self.canvas.create_image(0, 0, image=self.photo, anchor=NW)

        self.after(self.delay, self.update_frame)
```

f) Lanjutan source code GUI

```
407     def update_option_users(self):
408         self.list_user_names = get_list_user_names()
409         self.option_users_dict = get_option_user_dicts()
410         self.selected_user.set("-- Pilih --")
411         menu = self.option_users['menu']
412         menu.delete(0, 'end')
413         for user_name in self.option_users_dict:
414             menu.add_command(label=user_name, command=lambda value=user_name: self.on_change_selected_user(value))
415 
416     def add_new_user(self, user_name):
417         new_user = create_user(user_name)
418         if new_user:
419             is_agree_make_data = messagebox.askyesno(
420                 title="Buat dataset wajah",
421                 message="Sistem akan menonaktifkan absensi untuk membuat dataset wajah {0}. "
422                         "Pastikan wajah {0} tampak di kamera. Lanjutkan?".format(new_user.name))
423             if is_agree_make_data:
424                 if self.face_recognition_switch:
425                     self.face_recognition_switch.set(0)
426                     self.generate_dataset(user_id=new_user.id)
427 
428     def generate_dataset(self, user_id):
429         self.update_info_message("Membuat Dataset Wajah...")
430         generate_dataset_thread = GenerateDataset(user_id, self.video_comp)
431         generate_dataset_thread.daemon = True
432         generate_dataset_thread.start()
433         self.monitor_generate_dataset_thread(generate_dataset_thread)
```

```
435     def monitor_generate_dataset_thread(self, thread):
436         if thread.is_alive():
437             if thread.sample_number < total_sample:
438                 self.after(50, lambda: self.update_info_message(
439                     "Ekstrak frame : {0} dari {1}".format(thread.sample_number, total_sample)))
440             # check the thread every 50ms
441             self.after(50, lambda: self.monitor_generate_dataset_thread(thread))
442         else:
443             self.update_option_users()
444             self.after(100, lambda: messagebox.showinfo(
445                 title="Sukses",
446                 message="Dataset wajah berhasil dibuat. Silakan tekan tombol ON untuk mengaktifkan absensi."))
447 
448     def export_absensi(self, export_type):
449         if export_type == ExportType.USER.value:
450             selected_user = self.option_users_dict.get(self.selected_user.get())
451             if selected_user:
452                 list_data = get_list_absensi_user(selected_user)
453                 print(export_type, self.selected_user.get(), "Total: {0}".format(len(list_data)))
454                 self.generate_pdf(export_type, list_data, self.selected_user.get())
455         else:
456             selected_date = self.selected_date.get()
457             if selected_date:
458                 list_data = get_list_absensi_date(selected_date)
459                 print(export_type, self.selected_date.get(), "Total: {0}".format(len(list_data)))
460                 self.generate_pdf(export_type, list_data, selected_date)
461 
```

```
    def generate_pdf(self, export_type, list_data, keyword):
        file_name = "Export_{0}_{1}_{2}.pdf".format(export_type, keyword, datetime.now().strftime("%Y%m%d_%H%M%S"))
        doc = SimpleDocTemplate(file_name, pagesize=A4)
        elements = []
        content = []
        headers = ["No.", "Tanggal", "Nama", "Jam Datang", "Jam Pulang", "Status"]
        content.append(headers)
        for index, data in enumerate(list_data):
            number = index + 1
            date = data.date
            name = data.user.name
            check_in_time = data.check_in_time or ""
            check_out_time = data.check_out_time or ""
            status = data.status.value
            row = [number, date, name, check_in_time, check_out_time, status]
            content.append(row)

        t = Table(content)
        t.setStyle(TableStyle([
            ('INNERGRID', (0, 0), (-1, -1), 0.25, colors.black),
            ('BOX', (0, 0), (-1, -1), 0.25, colors.black),
        ]))
        elements.append(t)
        doc.build(elements)
        message = "Data berhasil di-export dengan nama file {0}".format(file_name)
        self.update_info_message("Export Finish")
        print(message)
        messagebox.showinfo(title="Sukses", message=message)


```

```
class VideoCaptureComp:
    def __init__(self, video_source):
        # Open the video source
        if type(video_source) is int:
            self.vid = cv2.VideoCapture(video_source, cv2.CAP_DSHOW)
        else:
            self.vid = cv2.VideoCapture(video_source)

        if not self.vid.isOpened():
            raise ValueError("Unable to open video source", video_source)

        # Get video source width and height
        self.width = self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)
        self.height = self.vid.get(cv2.CAP_PROP_FRAME_HEIGHT)

    def get_frame(self):
        if self.vid.isOpened():
            ret, frame = self.vid.read()
            if ret:
                # Return a boolean success flag and the current frame converted to BGR
                return ret, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            else:
                return ret, None
        else:
            return False, None

```

```
    # Release the video source when the object is destroyed
    def __del__(self):
        if self.vid.isOpened():
            self.vid.release()
            cv2.destroyAllWindows()

```

g) Source code model user

```python
from sqlalchemy import Column, String, Integer
from db.base_class import Base


class User(Base):
    id = Column(Integer, autoincrement=True, primary_key=True, index=True)
    name = Column(String(100), nullable=False)

    def __str__(self):
        return str(self.__dict__)

```

h) Source code model presensi

```python
from enum import Enum

from sqlalchemy import Column, Integer, ForeignKey, Enum as SQLEnum, Time, Date
from sqlalchemy.orm import relationship

from db.base_class import Base


class AbsensiType(str, Enum):
    CHECK_IN = "Datang"
    CHECK_OUT = "Pulang"


class AbsensiStatus(str, Enum):
    ABSENT = "Tanpa Keterangan"
    PRESENT = "Hadir"
    HOME = "Pulang"


```

```python
class Absensi(Base):
    id = Column(Integer, autoincrement=True, primary_key=True, index=True)

    user_id = Column(Integer, ForeignKey("user.id"))
    user = relationship("User", backref="list_absensi", lazy='subquery')

    type = Column(SQLEnum(AbsensiType), default=AbsensiType.CHECK_IN)
    date = Column(Date)
    check_in_time = Column(Time)
    check_out_time = Column(Time)
    status = Column(SQLEnum(AbsensiStatus), default=AbsensiStatus.PRESENT)

    def __str__(self):
        return str(self.__dict__)

```

i) Source code CRUD user

```python
from sqlalchemy.orm import Session

from db.models import User


def get_list_users(db: Session):
    return db.query(User).all()


def get_user(db: Session, user_id: int):
    return db.query(User).filter(User.id == user_id).first()


def create_user(db, name):
    new_user = User(name=name)
    db.add(new_user)
    db.commit()
    db.refresh(new_user)
    return new_user

```

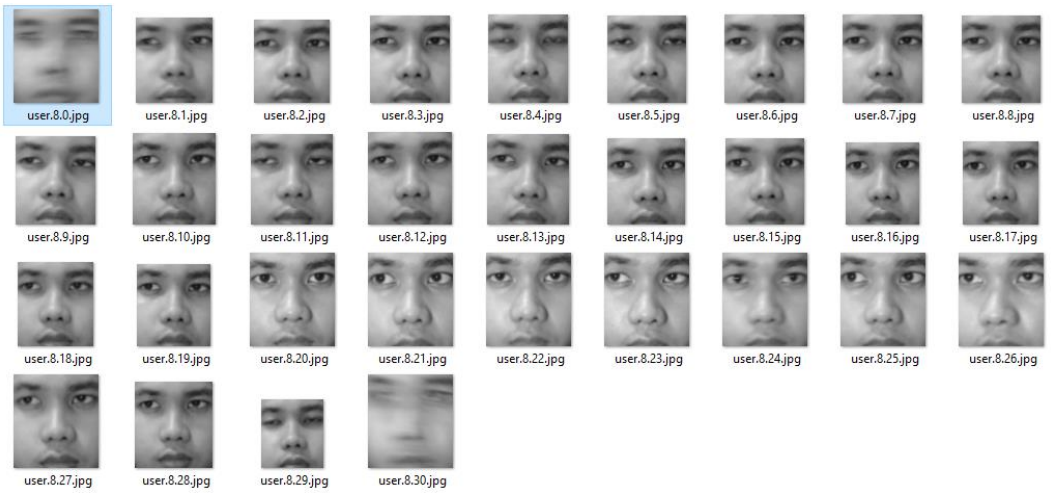
j) Source code CRUD presensi

```python
from datetime import datetime

from sqlalchemy.orm import Session

from db.models import Absensi
from db.models.absensi import AbsensiType, AbsensiStatus


def get_list_absensi(db: Session):
    return db.query(Absensi).all()


def get_list_absensi_by_user_id(db: Session, user_id: int):
    return db.query(Absensi).filter(Absensi.user_id == user_id).all()


def get_list_absensi_by_date(db: Session, date: str):
    return db.query(Absensi).filter(Absensi.date == date).all()


def get_absensi(db: Session, absensi_id: int):
    return db.query(Absensi).filter(Absensi.id == absensi_id).first()


def get_absensi_today_by_user_id(db: Session, user_id: int):
    current_date = datetime.now().strftime("%Y-%m-%d")
    return db.query(Absensi).filter(Absensi.user_id == user_id, Absensi.date == current_date).first()
```

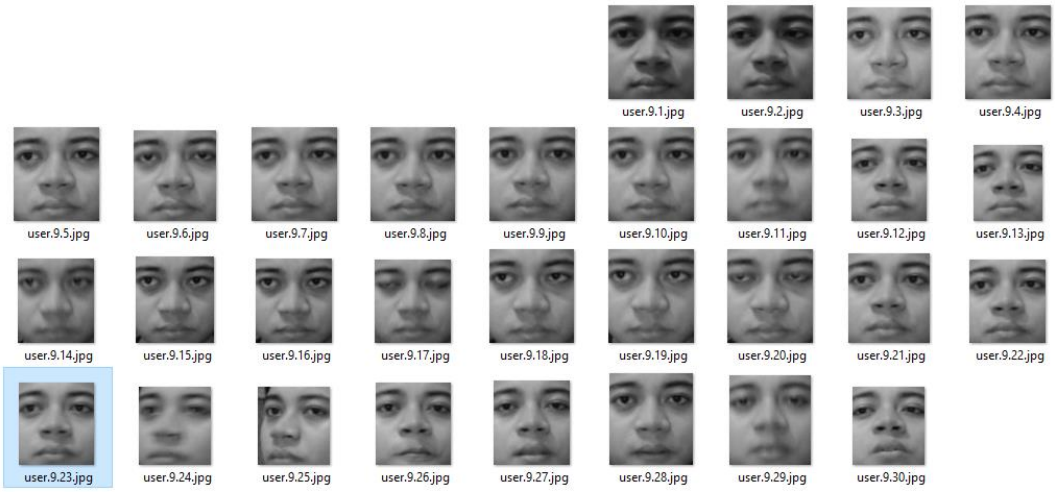
```python

def get_absensi_today_by_type_and_user_id(db: Session, type: AbsensiType, user_id: int):
    current_date = datetime.now().strftime("%Y-%m-%d")
    return db.query(Absensi).filter(Absensi.type == type, Absensi.user_id == user_id,
                                    Absensi.date == current_date).first()


def create_absensi(db: Session, user_id: int, type: AbsensiType, date: str, time: str):
    if type == AbsensiType.CHECK_IN:
        new_absensi = Absensi(user_id=user_id, type=type, date=date, check_in_time=time)
        db.add(new_absensi)
        db.commit()
        db.refresh(new_absensi)
        return new_absensi
    else:
        absensi = get_absensi_today_by_user_id(db, user_id)
        if absensi:
            db.query(Absensi).filter(Absensi.id == absensi.id).update({
                Absensi.type: type,
                Absensi.check_out_time: time,
                Absensi.status: AbsensiStatus.HOME
            })
            db.commit()
        return absensi

```

- **Data training**

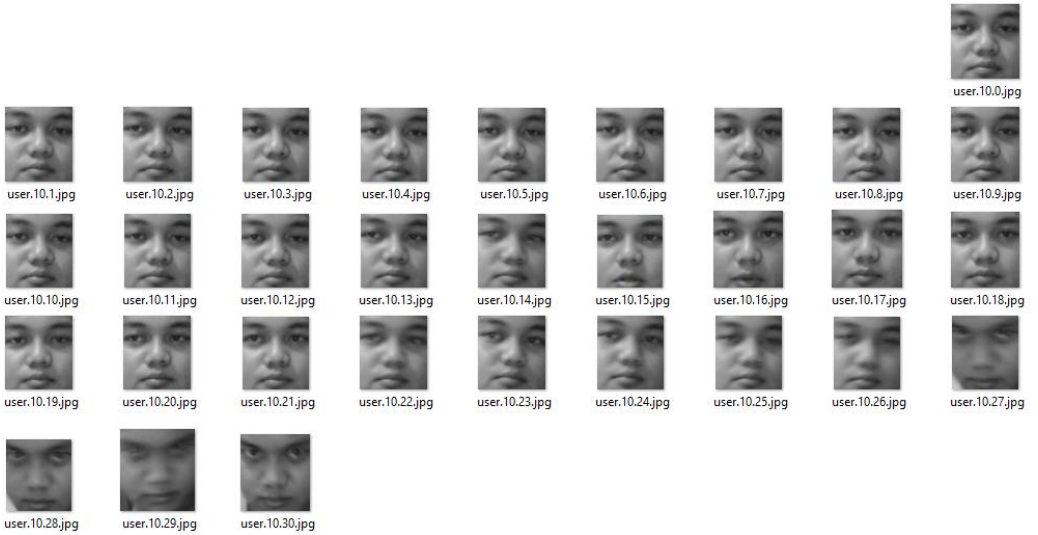
a) Data 0



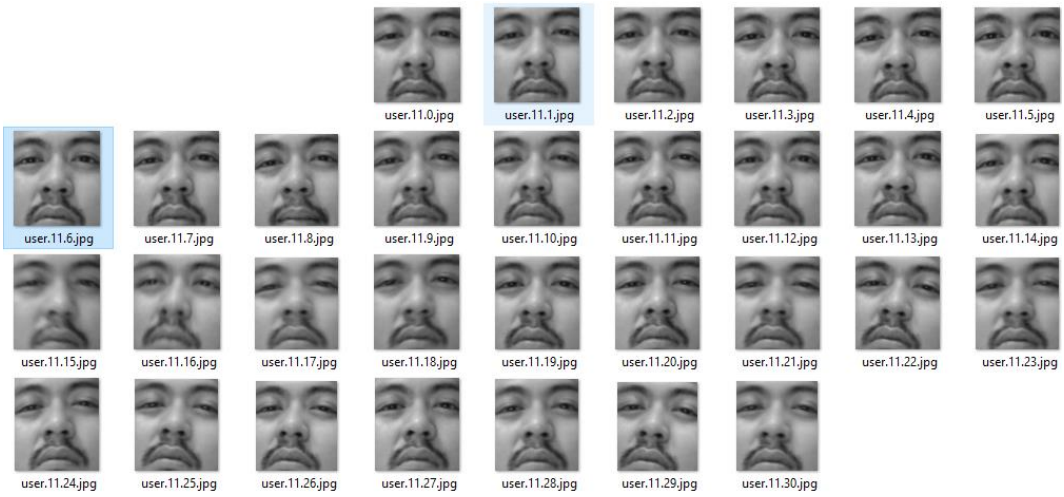
b) Data 1



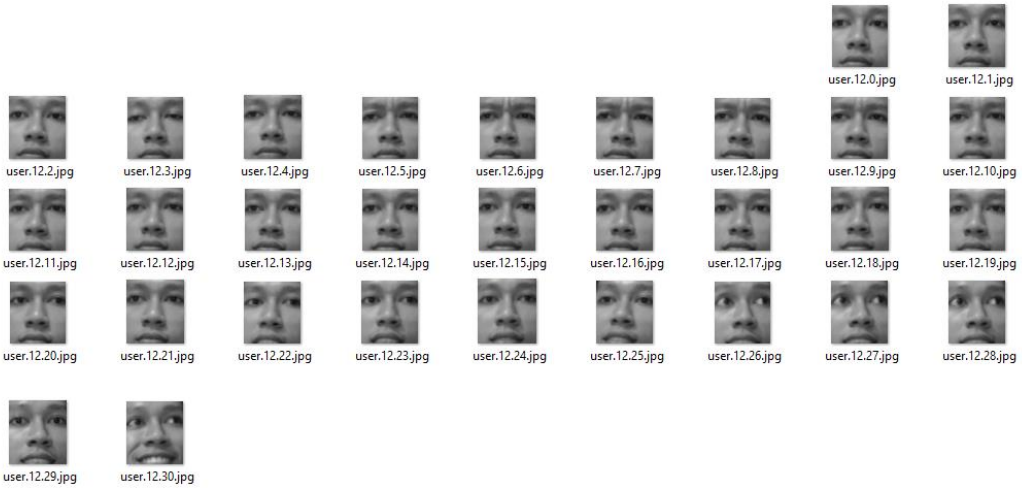
c) Data 2



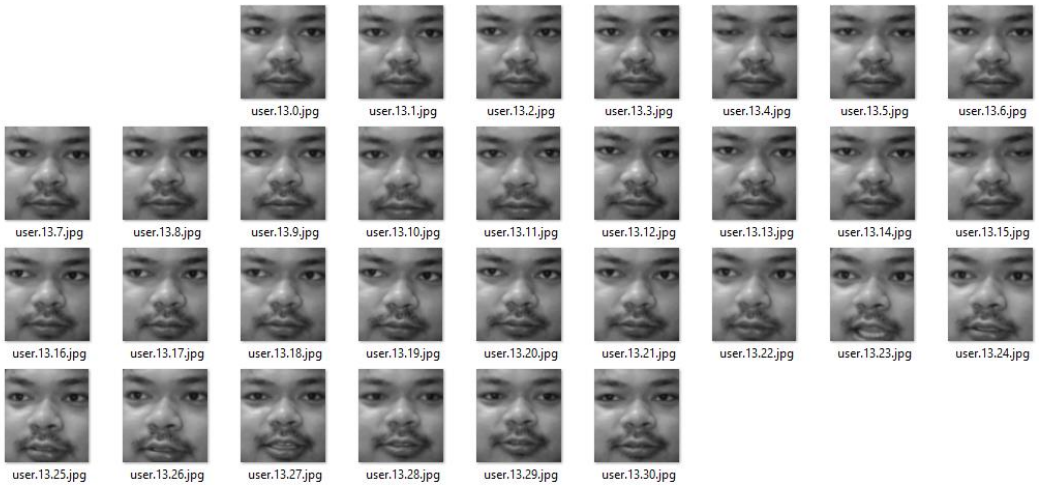
d) Data 3



e) Data 4



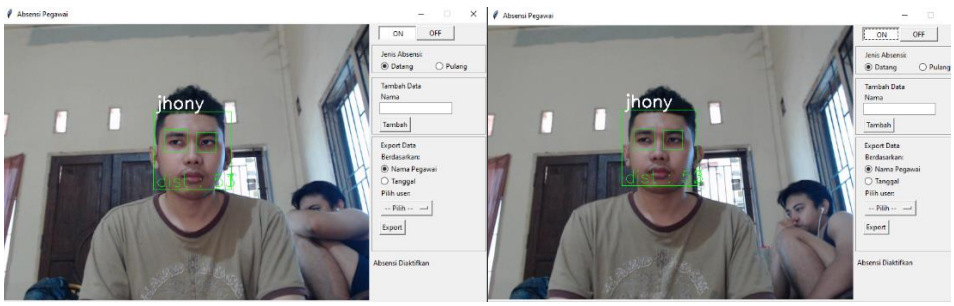
f) Data 5

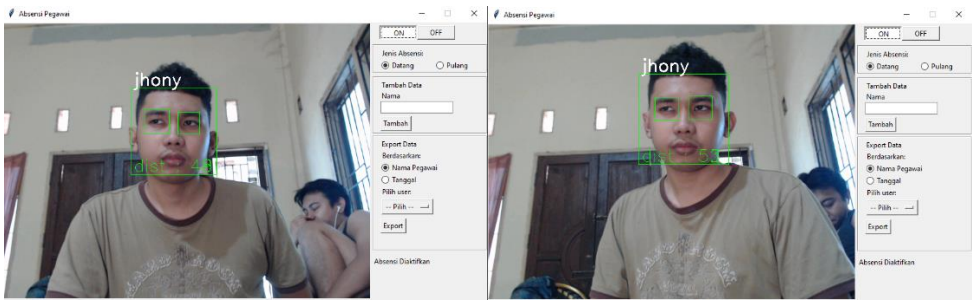


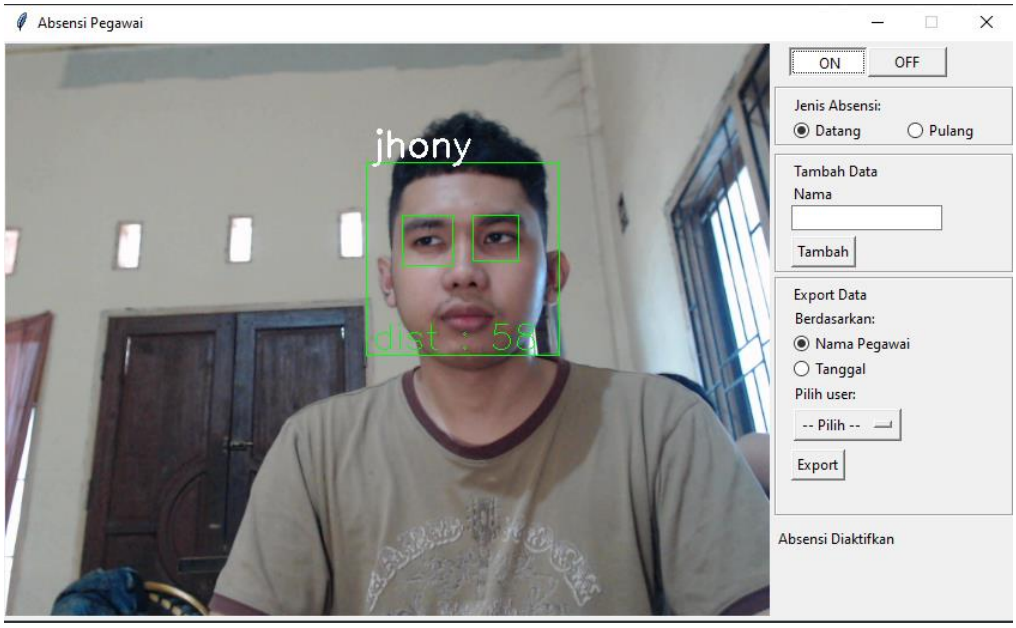
- **Data uji**

  a) **Data yang dilatih**

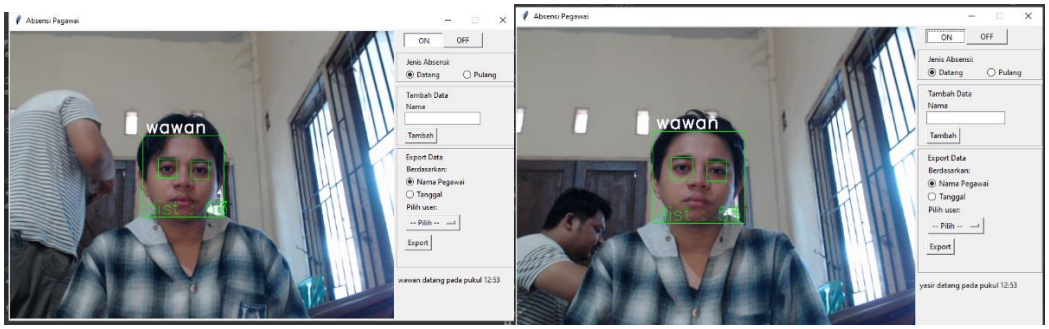
     1. **Data 0**

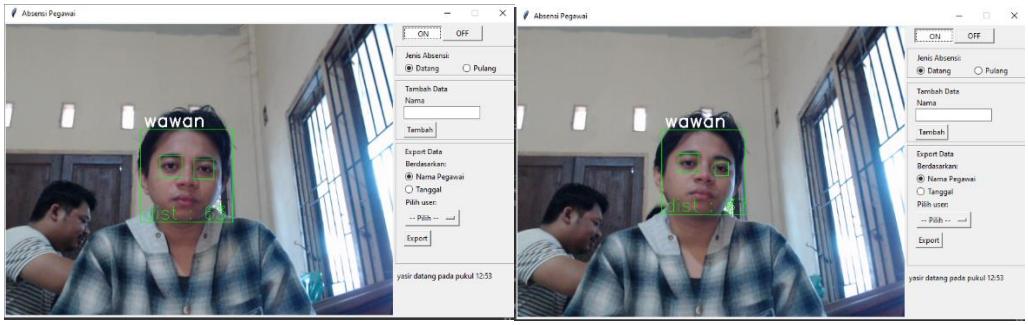


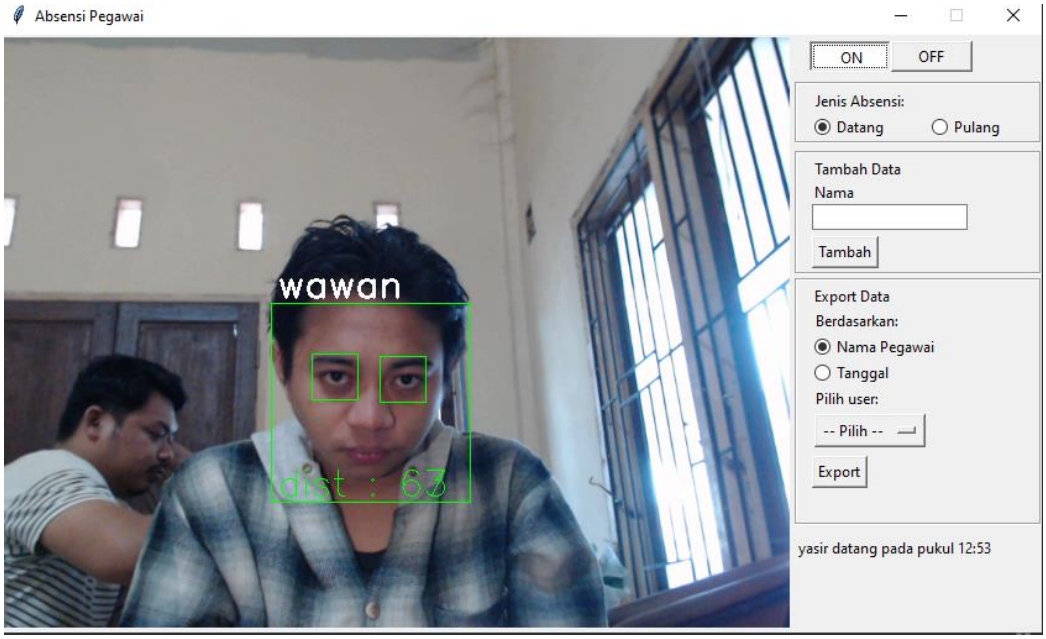




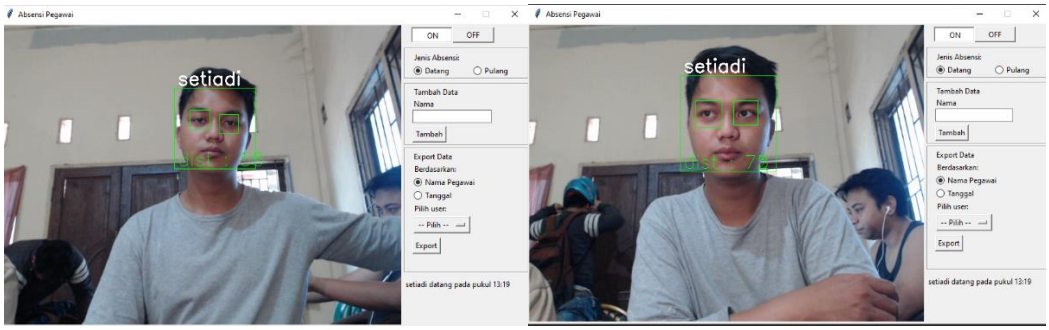
2. Data 1

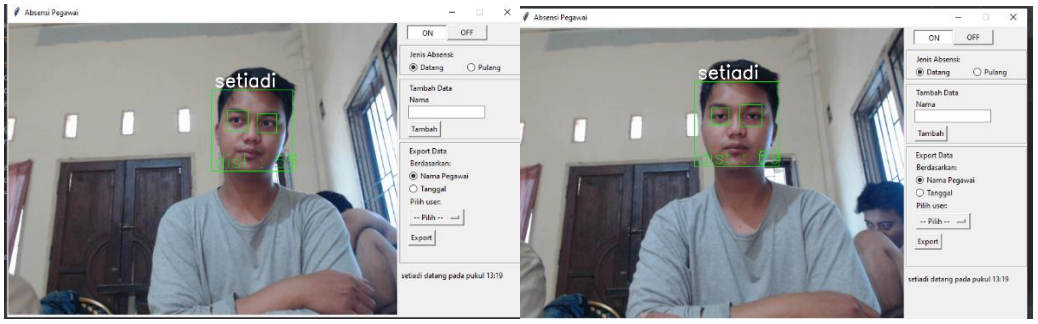

Absensi Pegawai
wawan
dist : 63
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
wawan datang pada pukul 12:53
yasir datang pada pukul 12:53

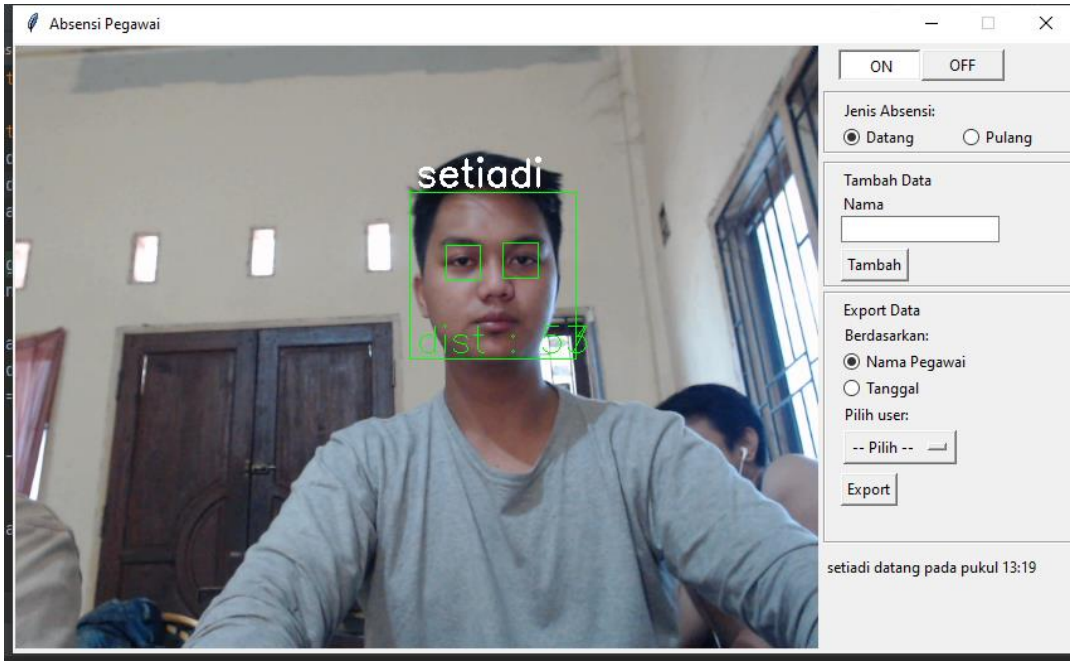

Absensi Pegawai
wawan
dist : 63
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
yasir datang pada pukul 12:53


Absensi Pegawai
wawan
dist : 63
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
yasir datang pada pukul 12:53

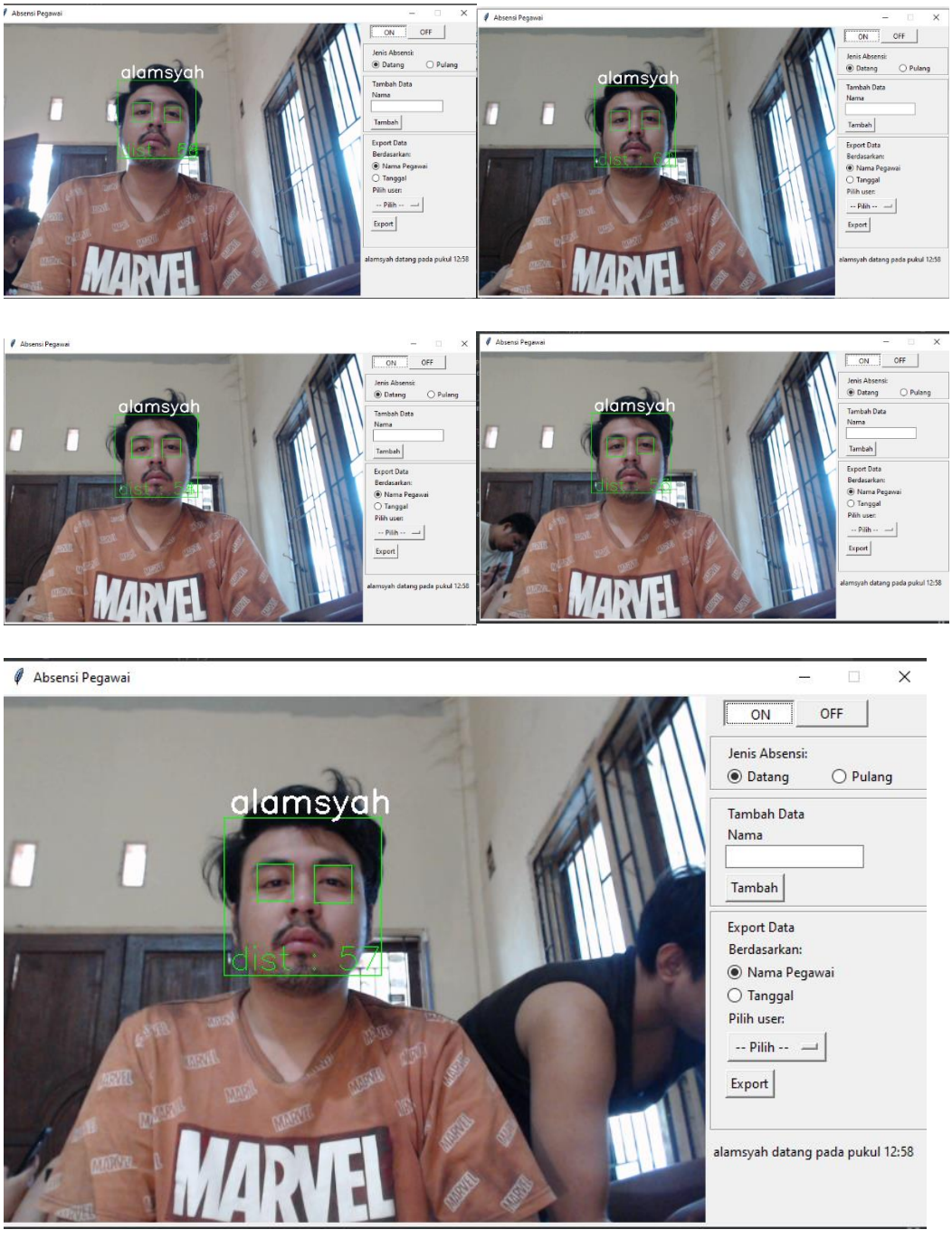
3. Data 2


Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
setiadi
setiadi datang pada pukul 13:19

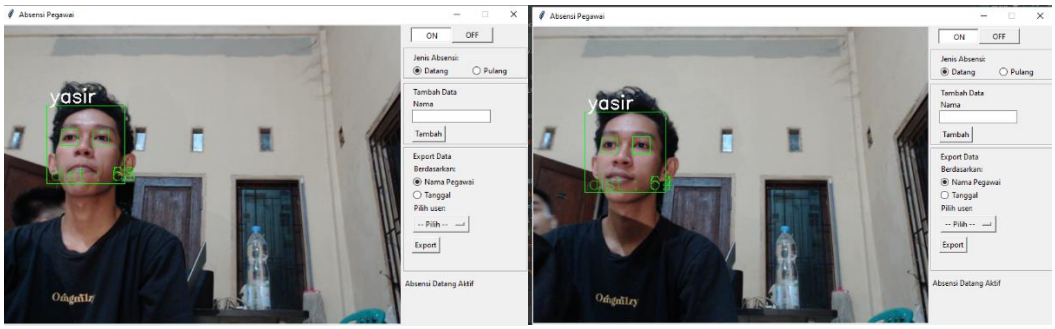

Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
setiadi
setiadi datang pada pukul 13:19

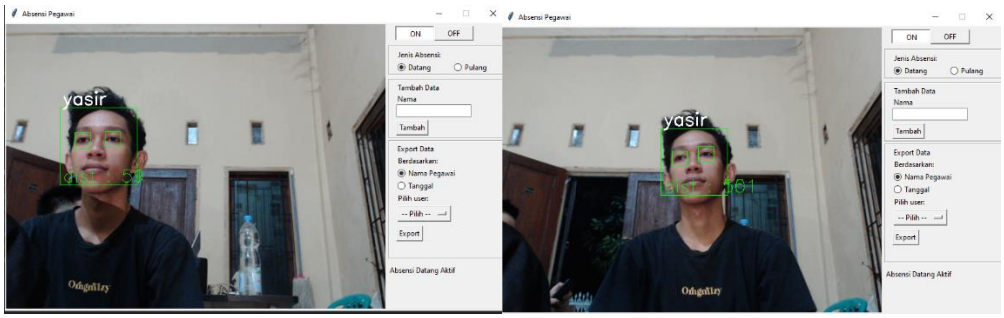

Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
setiadi
dist : 67
setiadi datang pada pukul 13:19

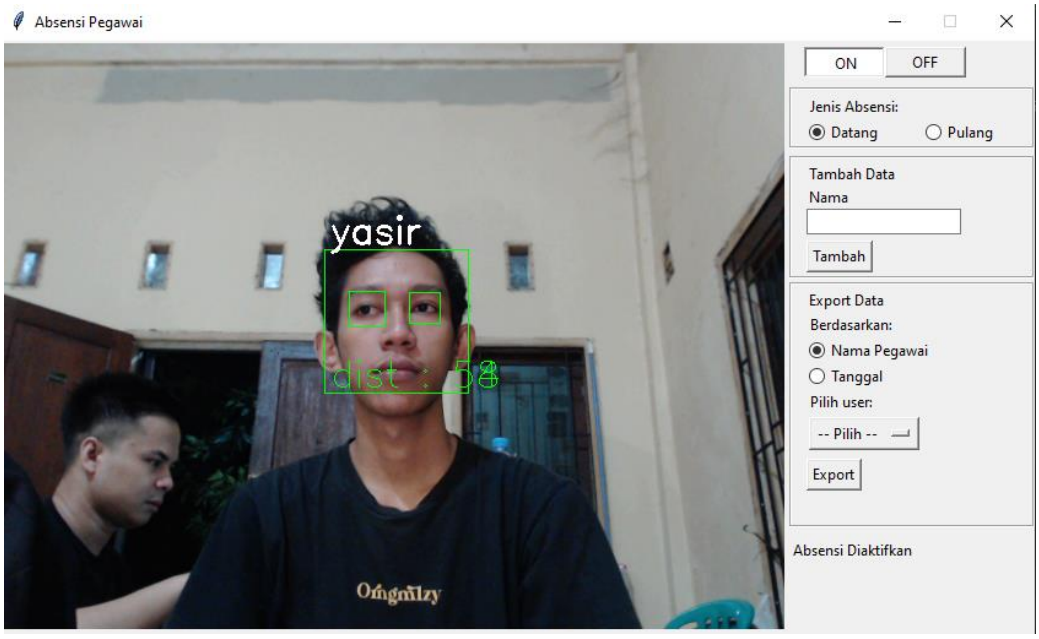
4. Data 3


Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
alamsyah datang pada pukul 12:58
alamsyah
dist : 57
MARVEL

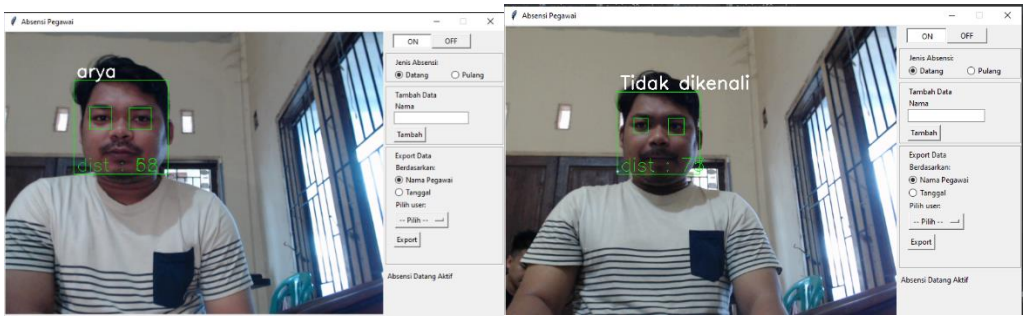
5. Data 4

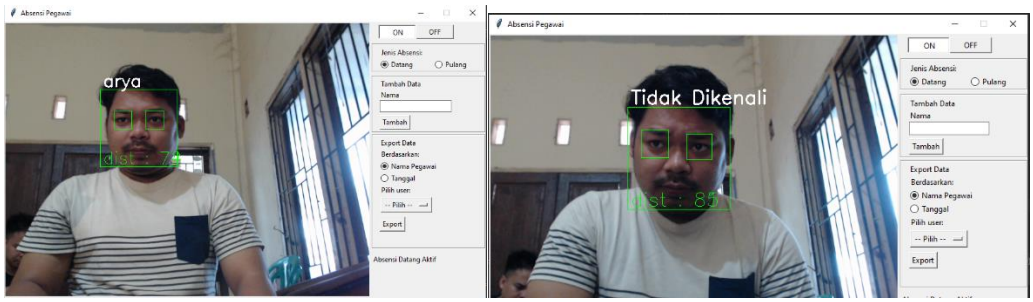

Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
yasir

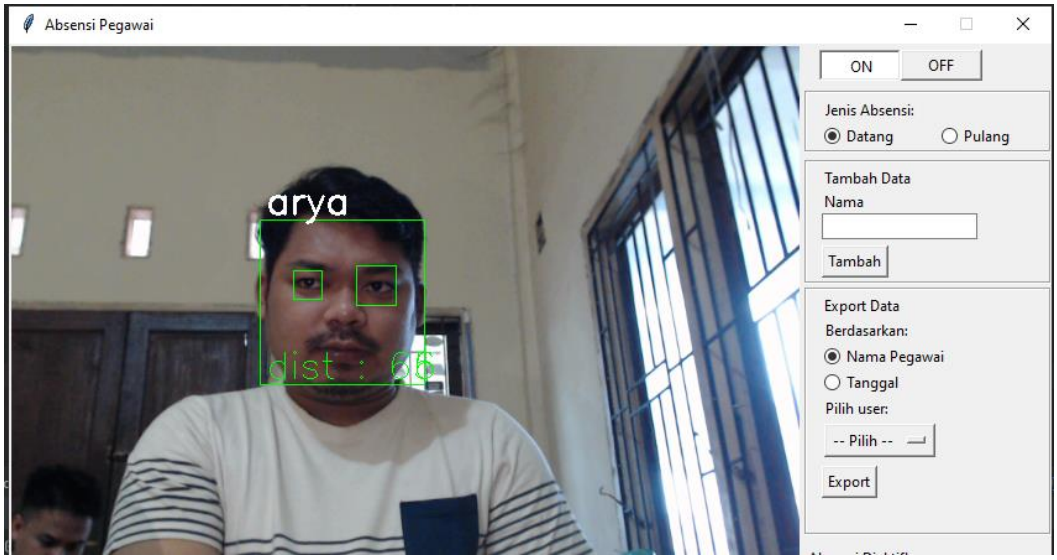

Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
yasir


Absensi Pegawai
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan
yasir
dist : 58

6. Data 5

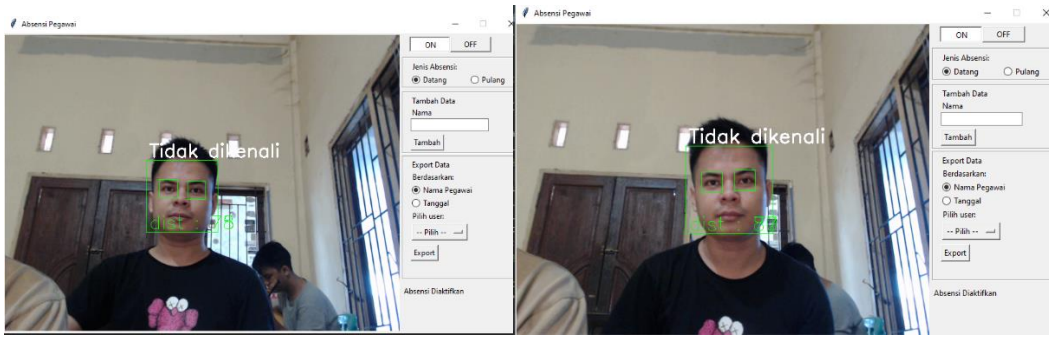

Absensi Pegawai
arya
dist : 68
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
Absensi Pegawai
Tidak dikenali
dist : 74
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif

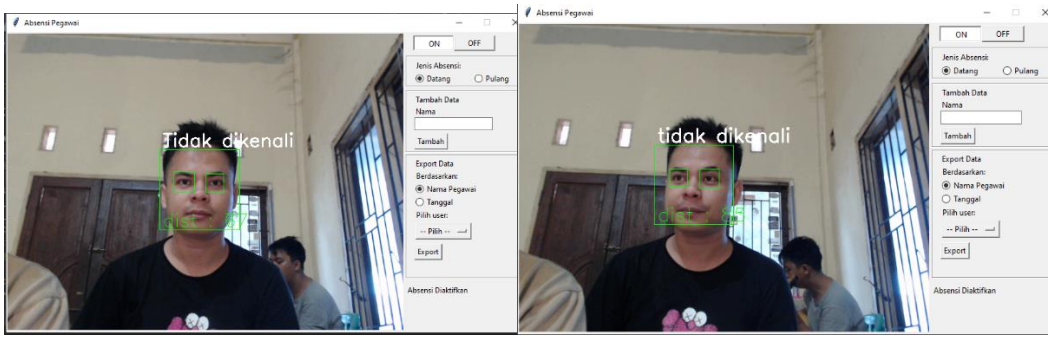

Absensi Pegawai
arya
dist : 74
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
Absensi Pegawai
Tidak Dikenali
dist : 85
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export

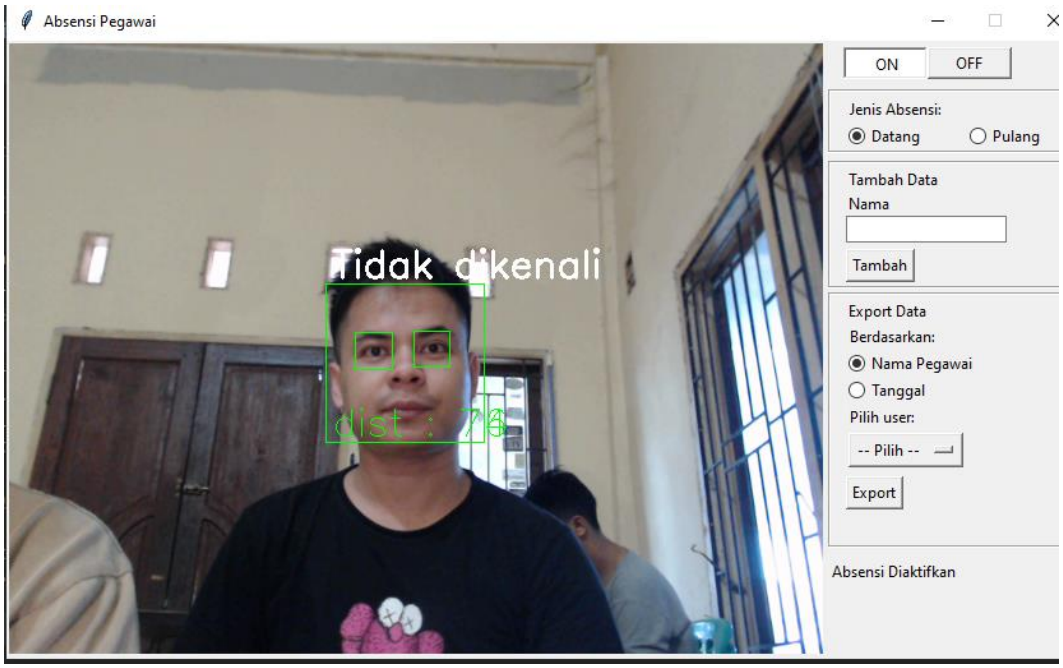

Absensi Pegawai
arya
dist : 66
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export

b) Data yang tidak dilatih

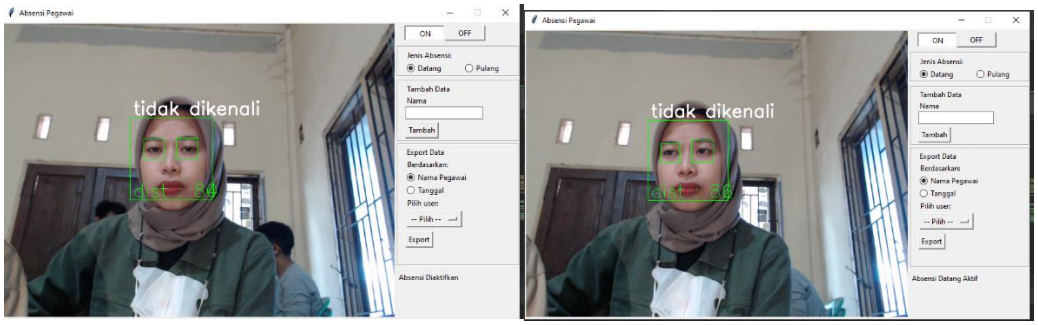
1. Data 0

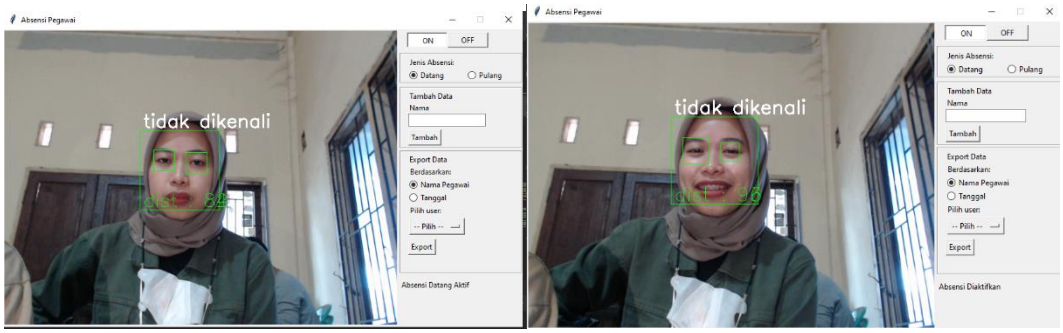


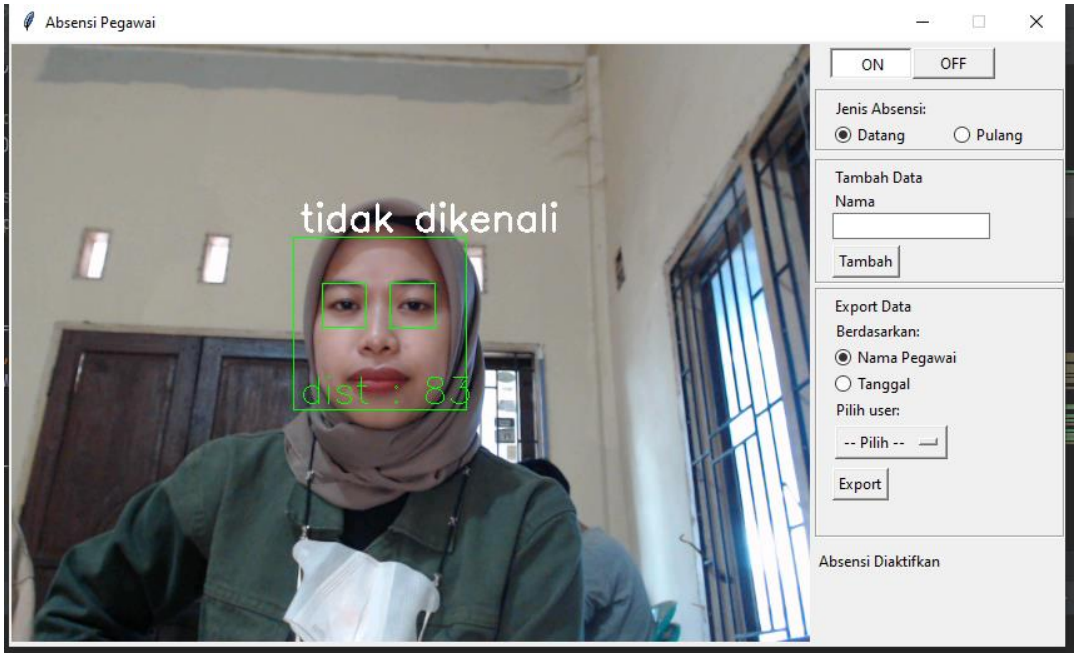




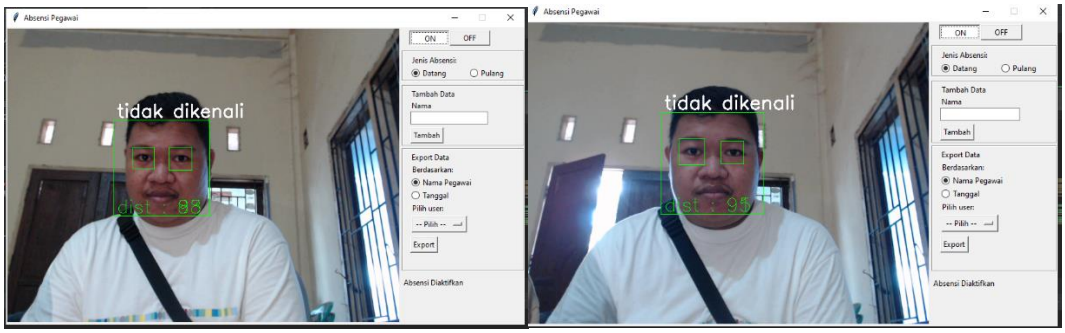
2. Data 1

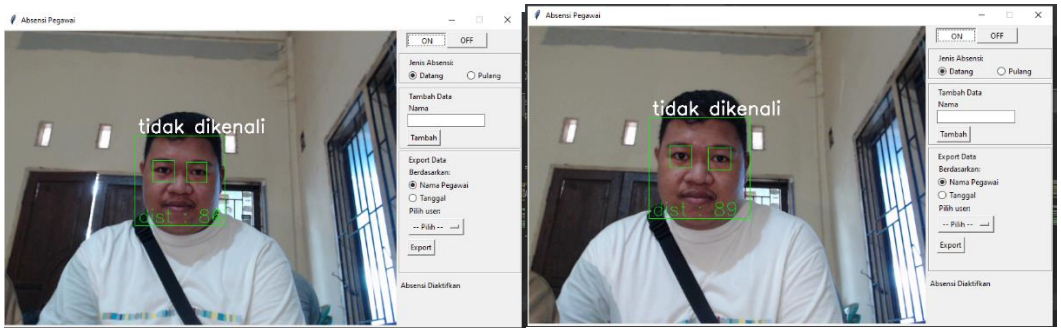

Absensi Pegawai
tidak dikenali
dist : 84
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan
Absensi Datang Aktif

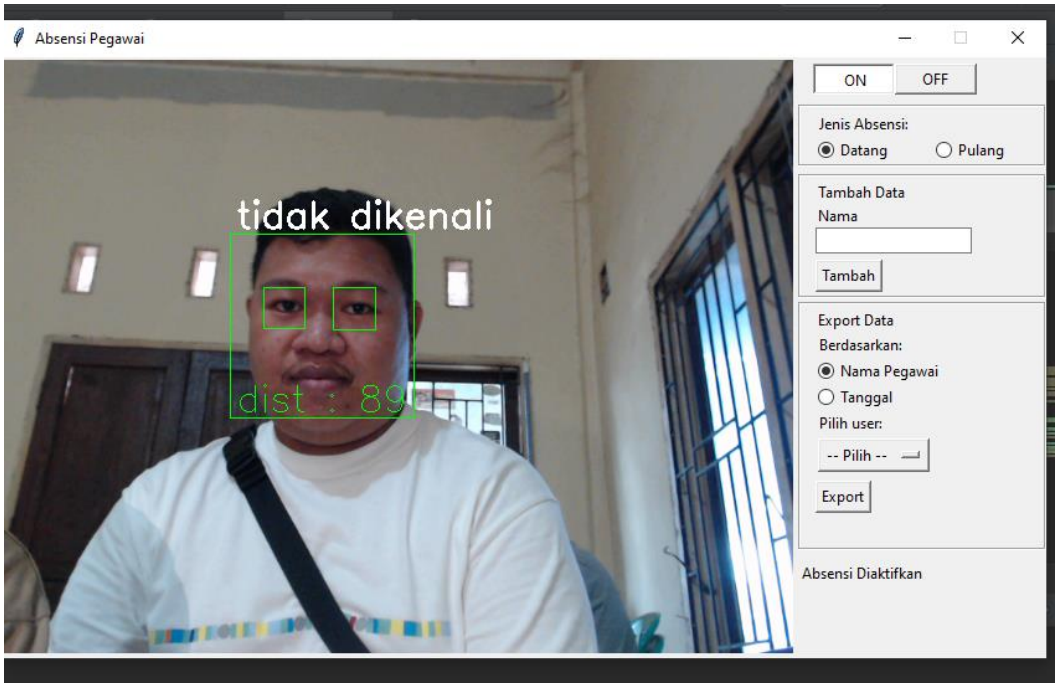

Absensi Pegawai
tidak dikenali
dist : 84
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Datang Aktif
Absensi Diaktifkan


Absensi Pegawai
tidak dikenali
dist : 83
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan

3. Data 2


Absensi Pegawai
tidak dikenali
dist : 95
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan


Absensi Pegawai
tidak dikenali
dist : 89
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan


Absensi Pegawai
tidak dikenali
dist : 89
ON
OFF
Jenis Absensi:
Datang
Pulang
Tambah Data
Nama
Tambah
Export Data
Berdasarkan:
Nama Pegawai
Tanggal
Pilih user:
-- Pilih --
Export
Absensi Diaktifkan