BAB III
LANDASAN TEORI

III.1. Pengenalan Wajah

Pada dasarnya, sistem pengenalan wajah tiga dimensi merupakan sistem pendeteksi untuk menentukan wajah seseorang dengan cara membandingkan sebuah citra wajah dengan model-model wajah yang telah disimpan sebelumnya (Heranurweni, 2010). Pengenalan wajah dapat diartikan sebagai pengelompokan sebuah wajah sebagai "wajah dikenali" atau "wajah tidak dikenali", setelah membandingkan ke dalam wajah individu dikenali yang tersimpan sebelumnya (database wajah). Terdapat 6 komponen utama dalam pengenalan wajah (Saputra, et al., 2013):

1. *Acquisition Module*
   Citra wajah dihadirkan ke sistem di modul ini melalui media input atau dari data di penyimpan.
2. *Pre-Processing Module*
   Hasil dari modul akuisisi akan dinormalisasi dan apabila diperlukan akan dilakukan pengolahan lanjutan agar didapatkan hasil pengenalan yang lebih baik.
3. *Feature Extraction Module*
   Modul ini bertanggung jawab terhadap penyusunan sebuah *feature* vector yang terbaik untuk mempresentasikan citra wajah.
4. *Classification Module*
   Fitur terektraksi dari sebuah wajah dapat dibandingkan dengan yang lainnya yang tersimpan di perpustakaan wajah (database wajah) dengan bantuan *pattern classifier*.

5. *Training set*

   Modul ekstraksi fitur dan modul klasifikasi menyesuaikan parameter mereka agar dapat mencapai performa pengenalan optimum dengan penggunaan *training set*.

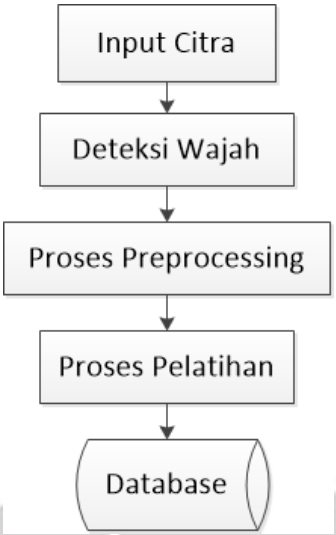
6. *Face library or face database*

   Modul ini menyimpan fitur vektor dari citra wajah di himpunan data pelatihan (training set)

III.2. Eigenface

Metode *Eigenface* yaitu membandingkan kode wajah di sebuah kamera dengan wajah yang telah tersimpan pada database. Database berisi beragam wajah yang telah dikodekan secara serupa. Algoritma pengenalan wajah dimulai dengan membuat matriks kolom dari wajah yang disimpan ke dalam database. Rata-rata vektor citra dari matriks kolom dihitung dengan cara membaginya dengan jumlah banyaknya citra yang disimpan di dalam database (Gurusinga & Arbi, 2013).

Untuk menyusun Flatvektor matriks citra wajah yang telah disimpan sebagai *image training* menjadi 1 matriks tunggal. Misalnya, citra dengan dimensi N x (H X W) adapun representasi semua matrks training menjadi matriks dalam bentuk N X 1 (Gurusinga & Arbi, 2013).

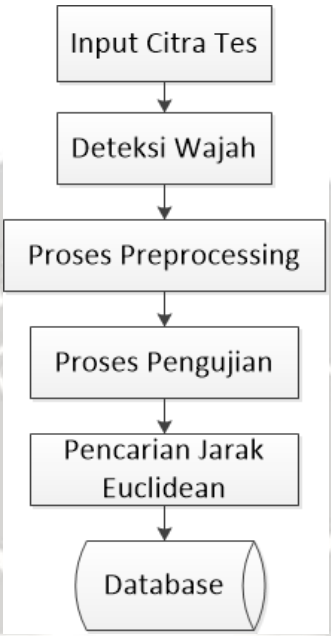


Gambar 3.1. Flowchart Pelatihan Eigenface

Pada gambar 2.1 terlihat bahwa diagram alir pelatihan *Eigenface*. Tahap pertama, wajah ditangkap kamera dan dideteksi. Tahap berikutnya adalah tahap *preprocessing* yang digunakan untuk normalisasi ukuran citra RGB ke *grayscale*, *histogram equalization* untuk memperbaiki kualitas citra input agar memudahkan proses pengenalan tanpa menghilangkan informasi utamanya. Dari proses tersebut menghasilkan citra wajah yang diambil dalam bentuk *png atau *jpg. Proses pelatihan pada metode *Eigenface*:

1. Buat Flatvektor dari setiap matriks image, di mana setiap *image* adalah W x H piksel.
2. Gabungkan setiap *image* dalam WH elemen vektor dengan menggabungkan semua baris. Buat ImageMatrix sebagai matriks N x WH berisi semua gambar yang digabung.
3. Jumlahkan semua baris pada ImageMatrix dan bagi dengan N untuk mendapatkan rataan gambar gabungan. Lalu namakan vektor elemen WH ini dengan R.

4. Kurangi ImageMatrix dengan *average image* R. Namakan matriks baru ukuran N x WH sebagai R'. (Proses perhitungan *Eigenface* data pelatihan).
5. Jika pada elemen-elemen dari matriks R' ditemukan nilai negatif, maka ganti nilainya dengan nilai 0.



Gambar 3.2. Flowchart Pengujian Eigenface

Pada gambar 2.2 terlihat bahwa diagram alir pengujian *Eigenface*. Tahap pertama, wajah ditangkap kamera dan dideteksi. Tahap berikutnya adalah tahap *preprocessing* yang digunakan untuk normalisasi ukuran citra RGB ke *grayscale*, *histogram equalization* untuk memperbaiki kualitas citra input agar memudahkan proses pengenalan tanpa menghilangkan informasi utamanya. Dari proses tersebut menghasilkan citra wajah yang diambil dalam bentuk *png atau *jpg. Proses pengujian pada metode *Eigenface* :

1. Buat Flatvector dari matrix *image* yang akan di tes, di mana *image* adalah W x H piksel.
2. Kurangi flatvector *image* yang akan di tes dengan *average image* R. Namakan matriks baru ukuran N x WH sebagai R'.

Selanjutnya mencari jarak Euclidian. Jarak Euclidian merupakan akar dari jumlah selisih kuadrat antara 2 vektor.

3. Mencari jarak euclidian dari flatvector *image training* dengan flatvector *image* tes.
4. Menjumlahkan elemen-elemen penyusun vektor dari setiap matriks yang diperoleh.

Mencari nilai terkecil dari hasil penjumlahan elemen-elemen penyusun vektor.

III.3. OpenCV

OpenCV merupakan singkatan dari *Open Source Computer Vision Library Opencv* dimulai di intel pada tahun 1999 oleh Gary Bradski dengan tujuan riset dan aplikasi komersial komputer di dunia (Emami & Suciu, 2012). OpenCV berisi beberapa fungsi C dan kelas C++, fungsi dan kelas tersebut diimplementasikan pada algoritma-algoritma *Image Processing* dan *Computer Vision* seperti pengenalan wajah, deteksi gerakan dan identifikasi objek.

OpenCV dikembangkan hingga sekarang dalam berbagai platform, termasuk Android. OpenCV didesain untuk aplikasi *real time* dan memiliki fungsi-fungsi akuisisi yang baik untuk gambar atau video (Arsy, et al., 2016).

III.4. Android

Android merupakan *Operating System (OS) mobile* yang tumbuh ditengah OS lainnya yang berkembang dewasa ini. Android pertama kali dikembangkan oleh perusahaan bernama Android Inc., dan pada tahun 2005 di akuisisi oleh raksasa Internet Google. Untuk setiap release-nya diberi kode nama berdasarkan nama hidangan makanan.

Android menyediakan platform terbuka bagi para pengembang untuk membuat aplikasi mereka sendiri untuk digunakan berbagai macam peranti bergerak (Listyorini & Widodo, 2013). Android SDK merupakan perangkat lunak untuk membuat dan mengembangkan aplikasi android. Di dalamnya terdapat library, debugger, android emulator, serta perangkat lunak lainnya yang dibutuhkan untuk membuat sebuah aplikasi android (Arsy, et al., 2016).