BAB II
TEORI PENUNJANG

2.1 Perkembangan Pengolahan Citra

Data atau informasi tidak hanya disajikan dalam bentuk teks saja, akan tetapi dapat berupa sebuah gambar, audio (bunyi, suara, musik), dan video. Keempat macam data atau informasi tersebut sering disebut multimedia. Citra/gambar merupakan salah satu komponen multimedia yang memiliki peranan penting sebagai informasi visual. Citra memiliki karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi. Ada sebuah peribahasa yang berbunyi "*a picture is more than a thousand word"* (sebuah gambar bermakna lebih dari seribu kata), maksudnya adalah sebuah gambar dapat memberikan informasi yang lebih banyak daripada informasi tersebut dalam bentuk kata-kata.

Citra secara harafiah, citra (*image)* adalah gambar pada bidang dwiwarna (dua dimensi). Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue)* dari intensitas cahaya pada bidang dwiwarna. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner)* dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam.

Pengolahan citra secara digital dimulai pada awal tahun 1921, yaitu pertama kalinya sebuah foto berhasil di transmisikan secara digital melalui kabel laut dari kota New York ke kota London (*Bartlane Cable Picture Transmission System)* dengan memakan waktu kurang dari 3 jam yang biasanya pengiriman suatu foto saat itu membutuhkan waktu sekitar satu minggu. Selanjutnya pada tahun 1960 tercatat perkembangan pesat dikarenakan munculnya teknologi komputer yang sanggup memenuhi suatu kecepatan proses dan kapasitas memori yang dibutuhkan oleh berbagai algoritma pengolahan citra.

Bidang ini sangat berhubungan erat dengan ilmu pengenalan pola, yang secara umum bertujuan mengenali suatu objek dengan cara mengekstrasi informasi penting yang terdapat pada suatu citra. Contoh aplikasi pengolahan citra pada berbagai disiplin ilmu :

1. Dalam bidang kedokteran terdapat sistem deteksi diagnosis suatu kelainan dalam tubuh manusia melalui citra yang dihasilkan oleh *scanner*.



Gambar 2.1 Hasil Pencitraan Menggunakan *Scanner*

1. Dalam bidang industri pengolahan citra di terapkan pada sistem pemeriksaan sauatu produk melalui kamera video.

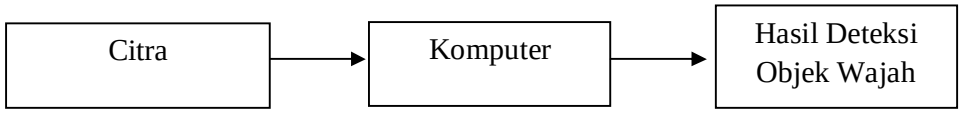
2. Dalam bidang perdagangan pegolahan citra digunakan untuk sistem mengelani huruf / angka dalam suatu formulir secara otomatis oleh mesin pembaca.

3. Dalam bidang militer sistem ini diterapkan pada pengenalan target peluru kendali melalui sensor visual.

2.2 Sistem Deteksi Wajah

Deteksi wajah adalah salah satu tahap *pra*-proses yang sangat penting di dalam sistem pengenalan wajah yang di gunakan untuk sistem *biometrik*. Deteksi wajah juga dapat digunakan untuk pencarian dan pengindeksan citra atau video yang didalamnya terdapat wajah manusia dalam berbagai ukuran, posisi, dan latar belakang.

Pendeteksian wajah merupakan proses segmentasi area wajah dengan latar belakang dari suatu citra masukan. Proses ini bekerja dengan cara memeriksa citra yang dimasukan apakah memiliki ciri wajah atau tidak. Jika memiliki ciri wajah, maka akan dilakukan proses pemisahan citra wajah dengan latar belakang citra yang dimasukan. Berikut ini merupakan diagram blok proses pendeteksian wajah:



Gambar 2.2 Diagram Blok Proses Pendeteksian Wajah

Proses pendeteksian pada diagram blok diatas dimulai dengan masukan berupa citra yang selanjutnya citra akan di proses oleh komputer dengan menggunakan *tools* yang tersedia dengan memanfaatkan metoda-metoda yang telah ada. Selanjutnya akan ditampilkan hasil dari proses deteksi wajah apakah terdeteksi atau tidak.

Masukan dapat berupa video ataupun citra, apabila masukan berupa video biasanya sistem berjalan secara *online* atau *real time* dan proses yang dapat dilakukan adalah proses *face tracking*. Sementara jika masukan berupa citra, biasanya sistem berjalan *offline*, maka dapat dilakukan proses pendeteksian wajah [19].

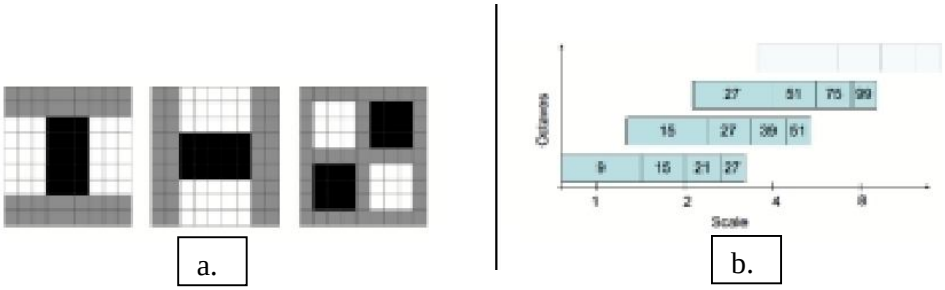
2.3 Metoda *Speed-Up Robust Features* (SURF)

Metoda *Speed-Up Robust Features (SURF)* merupakan sebuah metode deteksi fitur yang menggunakan *keypoint* dari sebuah citra/gambar. *Keypoint* itu sendiri adalah bagian-bagian dari sebuah citra/gambar yang nilainya kuat/tetap ketika mengalami perubahan skala, rotasi, *blurring*, transformasi 3 dimensi, pencahayaan dan juga perubahan bentuk.

Perubahan bentuk itu bisa terjadi karena bentuk citra data awal yang tidak utuh atau tidak sempurna gambar yang ada di dalam citra sampel tersebut. Citra uji yang tidak utuh mungkin karena ada objek lain yang menutupi, atau pengambilan gambar yang tidak sempurna, atau keadaan objek itu sendiri yang telah mengalami perubahan. Agar supaya *invarian* terhadap skala maka proses pertama yang dilakukan adalah membuat ruang sekala (*scale space).*

2.3.1 Ruang Skala (*Scale Space*)

Ruang skala di bagi kedalam bilangan *octave*, dimana satu *octave* mempresentasikan respon filter yang melakukan proses konvolusi citra masukan dengan filter yang ukurannya bertambah tinggi. Dalam SURF, level paling rendah dalam *scale space* diperoleh dari *output* filter 9x9. Filter ini berhubungan dengan nilai *Gaussian* σ =1.2. Ketika membuat filter yang lebih besar, ada beberapa faktor yang harus di perhitungkan terlebih dahulu. Kenaikan ukuran dibatasi oleh panjang *lobe* yang positif atau negatif dari turunan *Gaussian* orde kedua. Berikut ini presentasi dari turunan *Gaussian* orde kedua :



Gambar 2.3 (a) Turunan Orde Dua *Gaussian* (b) *Scale Scale*

Pada gambar 2.3(a) menunjukan *Lobe* positif yang mempresentasikan dengan warna kotak putih, sedangkan *lobe* negatif dipresentasikan dengan warna

kotak hitam. Gambar 2.3(a) menunjukan turunan orde dua *Gaussian* terhadap x,y dan xy.

Karena dalam proses ini kita membutuhkan posisi *central pixel* tetap, maka kenaikan ukuran *pixel* minimum 2 (satu *pixel* untuk setiap sisi), karena ada tiga *lobe* dimasing-masing filter adalah 6. Dengan demikian ukuran filter-filter untuk *octave* pertama adalah 9x9, 15x15, 21x21, 27x27. Sedangkan untuk setiap *octave* yang baru, ukuran kenaikan filter adalah *double* (dari 6 menjadi 12, menjadi 24, dan menjadi 48), sehingga ukuran filter *octave* kedua adalah 15,27,39,51. *Octave* ketiga 27,51,75,99, dan jika gambar asli masih lebih besar dari ukuran filter, maka *octave* keempat dibuat yaitu dengan ukuran filter 51,99,147 dan 195. Pada gambar 2.3(b) menunjukan contoh *scale space.*

2.3.2 Penempatan Lokasi *Keypoint* dengan Akurat

Untuk mencari lokasi *keypoint* yang *invariant* terhadap rotasi, maka dilakukan proses yang terdiri dari tiga langkah seperti di bawah ini :

1. Langkah pertama : Menentukan nilai ambang penentuan lokasi apakah *keypoint* atau tidak. Ketika nilai ambang dinaikan, jumlah *keypoint* yang terdeteksi lebiih kecil dan sebaliknya. Oleh karena itu, nilai ambang bisa disesuaikan untuk setiap aplikasi.

2. Langkah kedua : Sesudah proses penentuan ambang, selanjutnya proses *non-maxima suppression* dilakukan untuk mencari sekumpulan calon *keypoint*. Proses ini dilakukan dengan membandingkan masing-masing *pixel* gambar pada *scale space* dengan 26 tetangga, yang terdiri atas 8 titik di *scale* asli dan 9 di masing-masing *scale* diatas dan dibawahnya (total keseluruhan 9+8+9= 26). Hasil dari *non-maxima suppression* adalah *keypoint* yang dihasilkan gambar.

3. Langkah ketiga : Proses ini adalah proses terakhir dimana dalam mencari *keypoint* yaitu dengan menggunakan interpolasi dengan data yang dekat dengan *keypoint*. Ini dilakukan dengan mencocockan *quadratic* 3 dimensi

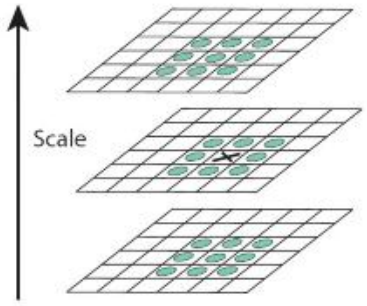
yang diajukan oleh Brown[3]. H(x, y, σ) menyatakan determinan *Hessian*, yang didefinisikan sebagai persamaan berikut ini :

$$H(\mathbf{x}) = H + \frac{\partial H^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 H}{\partial \mathbf{x}^2}\mathbf{x}$$

Lokasi ekstrim yang diinterpolasi, $\hat{x} = (x, y, \sigma)$, ditemukan dengan mencari turunan dari fungsi ini dan diberi nilai nol, yang dapat digambarkan dengan persamaan berikut ini :

$$\hat{x} = -\frac{\partial^2 H}{\partial \mathbf{x}^2}^{-1}\frac{\partial H}{\partial \mathbf{x}}$$

Jika $\hat{x}$ lebih besar dari 0.5 pada x,y atau arah σ, lokasi perlu disesuaikan lagi dengan melakukan interpolasi lagi. Prosedur ini diulang sampai nilai $\hat{x}$ kurang dari 0.5. Bila interpolasi telah dilakukan beberapa kali tetapi hasilnya tidak memenuhi nilai di bawah 0.5, maka *keypoint* tersebut dihapus. *Keypoint* yang tersisa adalah yang *stable.*



Gambar 2.4 *Non-Maxima Suppression*

2.3.3 Deskriptor *Keypoint* dan Proses *Matching*

Deskriptor ditentukan berdasarkan daerah piksel sekitar *keypoint.* Deskriptor ini menggambarkan distribusi intensitas piksel tetangga disekitar *keypoint*, sama halnya dengan informasi gradien yang di ekstrak oleh SIFT[4]. Pada SURF dihitung respon *Wavelet Haar* order pertama dalam arah x dan arah y

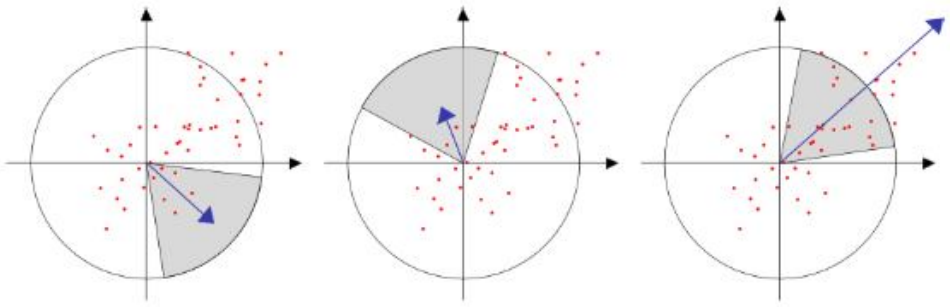
dan tidak menggunakan gradien seperti yang dilakukan pada metoda SIFT. Selain itu, SURF menggunakan integral gambar untuk meningkatkan kecepatan proses. Hal tersebut bertujuan untuk mengurangi waktu komputasi pada ekstraksi fitur dan pada proses *matching* fitur, yang juga telah terbukti meningkatkan *robust*[5].

Proses awal yang akan dilakukan adalah mencocokan orientasi yang dihasilkan berdasarkan informasi dari daerah yang berbentuk lingkaran disekitar piksel yang menjadi *keypoint*. Kemudian membuat daerah berbentuk kotak pada orintasi yang terpilih dan mengekstrak deskriptor SURF dari daerah tersebut. Pada proses selanjutnya, proses *matching* fitur antara dua gambar dilakukan. Berikut ini merupakan dua langkah penghitungan deskriptor *keypoint* yang akan dijelaskan secara mendetail.

2.3.3.1 Pembuatan Orientasi

Agar supaya *keypoint invarian* terhadap rotasi gambar, maka kita harus melakukan identifikasi orientasi yang dihasilkan disekitar *keypoint*. Untuk tujuan tersebut, langkah pertama yang dilakukan adalah dengan menghitung respon *Haar Wavelet* dalam arah x dan arah y pada daerah piksel tetangga disekitar *keypoint* yang berbentuk lingkaran yang dengan jarak 6*s*, dimana *s* merupakan *scale* dari *keypoint* yang terdeteksi. Ukuran *wavelet* ditentukan menjadi 4*s*. Filtering dipercepat prosesnya dengan integral gambar, dan keluaran yang dihasilkan direpresentasikan sebagai titik-titik dalam ruang dengan respon horizontal sepanjang sumbu *axis* dan respon vertikal sepanjang sumbu koordinat.

Orientasi dominan diperkirakan dengan menghitung jumlah semua respon dalam pergeseran orientasi dengan ukuran *window* π / 3 (seperti yang diunjukan pada gambar 2.5), dimana respon horizontal dan vertikal dalam *window* tersebut dijumlahkan. Dari kedua respon yang dijumlahkan akan menghasilkan sebuah vektor orientasi lokal. Vektor terpanjang mendefinisikan orientasi *keypoint*. Ukuran pergeseran *window* merupakan sebuah parameter yang harus dipilih secara hati-hati. Berikut ini merupakan gambar jendela pergeseran orientasi:



Gambar 2.5 Jendela Pergeseran Orientasi

2.3.3.2 Deskriptor Berdasarkan Jumlah Respon *Haar Wavelet*

Untuk proses ekstraksi deskriptor, langkah pertama yang dilakukan adalah membuat daerah kotak sekitar *keypoint*, dimana *keypoint* sebagai pusat dari daerah kotak tersebut, dan orientasinya di sekitar orientasi yang ditentukan. Berikut ini merupakan gambar jendela orienasi perhitungan deskriptor pada skala yang berbeda :

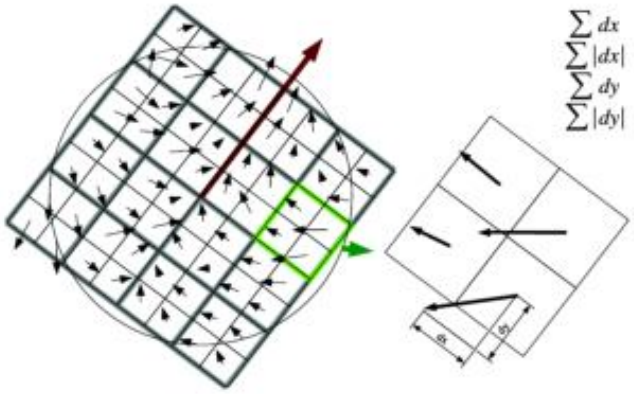


Gambar 2.6 Jendela Orientasi Perhitungan Deskriptor pada Skala yang Berbeda

Pada gambar diatas menjelaskan bahwa ukuran window yang diambil 20*s*, yaitu nilai terbaik yang diambil berdasarkan hasil eksperimen.

Pada tahap selanjutnya, respon *wavelet* dx dan dy dijumlahkan masing-masing sub-region. Selain memberikan informasi tentang polar dari perubahan intensitas, juga akan dihasilkan jumlah nilai absolut dari respon |dx| dan |dy|.

Masing-masing sub-region mempunyai empat dimensi deskriptor vertor *v*, yaitu *dx*, *dy*, |*dx*| dan |*dy*|. Untuk 4x4 sub-region, maka panjang vektor deskriptornya adalah 64 sebagaimana di ilustrasikan pada gambar beikut ini :



Gambar 2.7 perhitungan Deskriptor

Sementara untuk mendapatkan hasil vektor *v* dapat dilihat pada persamaan berikut ini :

$$v_{subregion} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right]$$

Respon *Wafelet* juga *invariant* terhadap pencahayaan, sedangkan sifat invarian terhadap kontras dicapai melalui pembentukan deskriptor ke dalam satuan vektor.

2.4 Deteksi *Interest Point*

Deteksi *interest point* merupakan sebuah terminologi baru dalam bidang *computer vision* yang mengacu pada *detection of interest point* untuk proses selanjutnya. Sebuah *interest point* adalah titik pada sebuah gambar yang secara umum dapat di golongkan sebagai berikut :

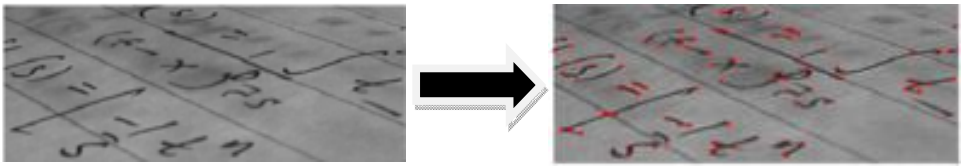
1. Memiliki kejelasan, terutama memiliki cukup alasan yang matematis, dan terdefinisi.

2. Memiliki posisi yang dapat difenisikan dengan baik dalam ruang gambar.

3. Struktur gambar di sekitar titik *interest point* kaya akan informasi (misalnya :tekstur 2D yang signifikan), sehingga penggunaan *interest point* dapat menyederhanakan/mempermudah sebuah proses pada *vision system.*

4. Titik ini stabil dibawah gangguan lokal dan global, dalam domain gambar sebagai variasi sedemikian rupa kecerahan(*iluminasi).* Sehingga, titik *interest point* dapat diandalkan untuk selanjutnya akan di hitung dengan reproduktifitas yang tinggi.

5. Opsional, pendapat tentang *interest point* harus menyertakan atribut skala, untuk memungkinkan menghitung *interest point* dari gambar/citra asli serta dibawah perubahan skala.



Gambar 2.8 Deteksi *Interest Point*

2.5 *Region Of Interest* (ROI)

Sebuah "*region of interest*" adalah bagian dari citra yang ingin disaring (*filter)* untuk membentuk beberapa operasi terhadapnya. Tujuan dari pemotongan ini adalah untuk mengambil citra wajah dan membuang citra lain yang tidak diperlukan. Dimensi citra yang dipotong disesuaikan dengan dimensi dari proses segmentasi atau pengkodean objek wajah yang dilakukan pada proses pendeteksian wajah.

ROI memungkinkan untuk mengakses bagian tertentu dari sebuah citra untuk diolah secara berbeda. Fitur ini menjadi sangat penting apabila terdapat bagian area tertentu dari citra yang dianggap lebih penting dari area lainnya.

2.5 *Grayscaling* Citra

Grayscaling citra merupakan tahapan pertama dari proses penyelarasan, pada ahap ini terjadi pengkonversian citra warna RGB menjadi citra berwarna keabu-abuan. Citra warna RGB terdiri dari 3 parameter warna yaitu merah (*red)*, hijau *(green)* dan biru *(blue)*, jika citra warna RGB ini dimasukan ke dalam proses ekstraksi, maka proses tersebut akan sulit untuk dilakukan karena citra RGB terdiri dari 3 parameter, oleh karena itu diperlukan penyamaan parameter yaitu dengan melakukan tahap *grayscaling* ini. Berikut ini adalah persamaan tahap *grayscaling* citra :

$$X = 0.299r + 0.587g + 0.114b$$

Dimana, nilai citra RGB adalah (r,g,b) dengan nilai *integer* antara 0 sampai 255, dan x adalah nilai *grayscale.*

2.6 Modul *Matching* Fitur SURF Menggunakan Metode FLANN

Metode *Fast Library Approximated Nearest Neighbor (FLANN)* adalah sebuah *library* untuk melakukan pencarian cepat, perkiraan tetangga, yang terdapat pada *space* dimensi yang tinggi. *Libarary* ini merupakan kumpulan algoritma yang bekerja dengan baik untuk menemukan nilai tetangga terdekat, sementara untuk hasil parameter yang optimal, tergantung pada kumpulan data yang digunakan.

Metoda *Fast Library Approximated Nearest Neighbor (FLANN)* digunakan untuk *matching* fitur SURF citra wajah data awal dengan fitur SURF citra wajah uji. Fitur SURF ini terdiri dari 3 komponen utama, yaitu: *keypoint, deskriptor* dan vektor. Untuk satu gambar wajah citra uji terdapat *cluster* untuk fitur SURF. *Cluster* ini akan otomatis dengan menggunakan *K- Nearest Neighbor* (KNN) dengan tipe indeks *kd tree*, dimana KNN ini akan mencari jarak yang paling kecil antara vector sampel dengan vector pada *cluster*.

Proses *matching* fitur citra wajah uji dan fitur citra wajah sampel, vektor *keypoint* dan deskriptor pada citra wajah akan di cocokan nilainya dengan menggunakan KNN *search*. KNN *search* mencari *cluster* pada citra wajah uji yang nilai vektor deskriptornya paling dekat jaraknya dengan vektor deskriptor

pada citra wajah sampel. Setelah *cluster* diperoleh, akan dicari nilai vektor deskriptor citra uji pada *cluster* teresebut yang sama atau yang paling dekat dengan vektor deskriptor citra wajah sampel.

Jika terdapat yang sama, maka ada satu *keypoint* yang cocok antara kedua citra wajah tersebut. Selanjutnya, untuk masing-masing gambar akan ditarik suatu garis dari satu titik ke titik lainnya yang memiliki nilai *keypoint* yang cocok.

2.7 Perangkat Keras

Perangkat yang digunakan untuk pemrosesan adalah komputer / laptop, pemrosesan pendeteksian wajah pada penelitian ini memiliki spesifikasi sebagai berikut :

1. Jenis Laptop : Toshiba Satelite U-400
2. Sistem Operasi : Windows 7 64 Bit
3. Prosesor : Intel Core 2 Duo T6400 @2.00Ghz 2.00Ghz
4. Memori : 2GB DDR2
5. LCD : 13.3 inchi (Resolusi 1280 x 800)

2.8 Perangkat Lunak

Perangkat lunak yang digunakan pada penelitian ini merupakan perangkat lunak yang di gunakan untuk memproses citra masukan yang hasilnya merupakan dapat mempresentasikan apakah terdapat wajah atau tidak pada pendeteksian yang dihasilkannya.

2.8.1 Bahasa Pemrograman C++

Bahasa pemrograman C++ adalah suatu bahasa pemrograman komputer yang telah diperluas dengan kemampuan *object oriented*. Sejak tahun 1990, C++ merupakan bahasa pemrograman komersil yang paling populer. C++ dikembangkan oleh Bjarne Stroustrup pada tahun 1979 dari bahasa pendahulunya yaitu bahasa C.

Pada awal penciptaannya bahasa pemrograman C++ sering di sebut dengan istilah "C *with Classes*" dikarenakan terdapat sedikit perbedaan dengan bahas C, dimana perbedaan tersebut terletak pada penambahan kelas-kelas, fungsi virtual, operator *overloading*, *multiple inheritance*, *template*, penanganan eksepsi dan fitur-fitur lainnya. Telah banyak aplikasi-aplikasi yang dikembangkan, diantaranya perangkat lunak sistem, perangkat lunak aplikasi, dan lain-lain. Adapun kelebihan yang dimiliki bahas C++ ini adalah :

1. Bahasa C++ tersedia di semua jenis komputer.
2. Kode bahasa C/C++ sifatnya *portable* dan fleksibel untuk semua jenis komputer.
3. Proses eksekusi program bahasa C/C++ lebih cepat.
4. Dukungan pustaka yang tersedia sangat banyak.
5. Bahasanya terstruktur.
6. Sudah mendukung pemrograman berorientasi objek.

Adapun kekurangan yang dimiliki oleh bahasa pemrograman C++ adalah :

1. Banyaknya operator serta fleksibilitas penulisan program yang masih membuat bingung pemakai program.
2. Bagi pemula akan mengalami kesulitan ketika mencoba menggunakan *pointer* dan penerapan konsep pemrograman berorienasi objek.

Berikut ini merupakan contoh pemrograman C++ sederhana :

```
#include <iostream>

int main()

{

std:: cout<<"Hello,new world!\ n";

}
```

Keterangan:

1. *#include <iostream>*

   Pernyataan *#include* memberitahukan *pre-processor* untuk menyertakan kode dari *iostream*. Berkas *iostream* berisi deklarasi untuk berbagai fungsi yang dibutuhkan oleh perangkat lunak atau kelas-kelas yang dibutuhkan. *Pre-processor* adalah suatu program yang dijalankan oleh kompiler C++, yang memiliki kemampuan untuk menambahkan dan menghapus kode dari sumber.
2. *int main()*

   Pernyataan ini mendelekrasikan fungsi utama, fungsi adalah modul yang berisi kode-kode untuk menyelesaikan masalah-masalah tertentu.
3. *{*

   Kurung kurawal buka menandakan awal program.
4. *std:: cout<<"Hello,new world!\ n";*

   Cout adalah sebuah obyek dari *library* perangkat lunak standar C++ yang digunakan untuk mencetak *string* ke piranti *output* standar, biasanya layar komputer. Kompilator menghubungkan kode dari *library* perangkat lunak standar dengan kode yang telah ditulis untuk mendapatkan hasil yang dapat dieksekusi.
5. *\n*

Bentuk *modifier* yang digunakan untuk berganti baris setelah menampilkan *string.*

6. }

Kurung kurawal tutup menandakan akhir program.

2.8.2 Qt Creator

Qt *Framework* sudah sejak lama digunakan untuk mengembangkan aplikasi lintas *platform*. Qt sendiri dibuat pada tahuan 1996 oleh perusahaan asal Swedia yang bernama Troltech. Karena sifatnya yang lintas *platform,* kita dapat membuat aplikasi yang berjalan diatas *platform* Windows, Linux, dan Max. Qt *Framework* sudah didesain sedemikian rupa sehingga mudah digunakan oleh depelover tanpa harus mengorbankan fleksibilitas dan efesiensi. Qt mendukung pengembangan dengan dua bahas utama yaitu Objek *Oriented* C++ dan Java.

Qt *Framework* memiliki koleksi *class library* yang lengkap dan konsisten didukung oleh dokumentasi yang komprehensif. *Class library* tersebut berisi semua *function* yang dibutuhkan untuk mengembangkan aplikasi. Qt *Framework* membantu mengurangi pekerjaan developer dengan meningkatkan produktifitas penulisan kode untuk pengembangan yang bersifat *Rapid Applicatiom Development (RAD).*



Gambar 2.9 Qt Creator

Qt Creator adalah sebuah *Cross-platform* C++, yang memiliki *Integrated Development Environment (IDE)* yang merupakan sebagian dari Qt SDK. Qt

Creator sendiri merupakan sebuah *Visual Debugger* dengan mendukung tampilan *Grephical User Interface (GUI)* dan *form designer*. Editornya juga memiliki fitur berupa *higligting sintaks* dan *auto completion.* Qt Creator menggunakan kompiler C++ dan GNU *Compiler Collection on Linus* dan *FreeBSD.* Pada *Windows* kompilernya dapat menggunakan *MingGW* atau *MSVC* dengan pengaturan *default* dan dapat juga menggunakan *CDB* di*compile* dari sumbernya. Qt Creator merupakan aplikasi *open source* yang dapat digunakan secara *free*, sehingga banyak yang beralih ke penggunaan Qt Project, karena selain dasar pemrogramannya adalah C atau C++, Qt Creator merupakan *multi platform*, yang dapat digunakan di Windows ataupun di Linux.

2.8.3 *Open Source Computer Vision* (OpenCV)

Open Source Computer Vision adalah *library open source multi-platform* yang berlisensi *Berkeley Software Distribution* (BSD) yang bersifat *open source* sehingga bebas digunakan untuk hal-hal yang bersifat akademis dan penelitian. *Library* ini berfungsi untuk mentransformasikan data dari citra diam atau kamera video ke salah satu keputusan atau representasi baru. Semua representasi tersebut dilakukan untuk mencapai beberapa tujuan tertentu.



Gambar 2.10 OpenCV

OpenCV dikembangkan oleh *Visual Interactivity Group* di *Intel's Microprocessor Research Lab* pada tahun 1999. Tujuan dikembangkannya OpenCV adalah untuk membangun suatu komunitas *open source* dan

menyediakan situs yang mendistribusikan upaya dari komunitas sehingga dapat dikonsolidasi dan dioptimalkan dari segi kinerjanya.

Telah banyak penambahan kelas OpenCV, sehingga OpenCV sampai saat ini telah memiliki lebih dari 500 fungsi. Berikut ini beberapa fungsi yang dimiliki oleh OpenCV :

1. Metoda Geometrik
2. Pengenalan Objek
3. Pengukuran
4. Segmentasi
5. *Utilitas*
6. Fitur
7. Transformasi Citra
8. Piramida Citra
9. *Tracking*
10. Kalibrasi kamera
11. Pengolahan citra

Alasan penggunaan OpenCV karena kostumisasi yang dimiliki oleh OpenCV yang tinggi dan kinerja yang baik pada program berbasis bahasa *native C/C++*. Selain itu juga terdapat beberapa kelebihan OpenCV, diantaranya :

1. OpenCV gratis, karena *open source library.*
2. Memliki dokumen yang cukup banyak.
3. Dapat bekerja lebih cepat pada presesor berbasis Intel.

Selain kelebihan, OpenCV juga memiliki kekurangan, yaitu OpenCV kurang mendukung bahasa berbasis .NET seperti C#, Visual Basic dan lainnya,sehingga dibutuhkan *wrappers* mendukungnya. Pada penelitian ini menggunakan OpenCV, karena *tools* yang dimiliki oleh OpenCV dapat mendukung untuk penelitian deteksi wajah menggunakan metoda SURF ini.