

BAB II

LANDASAN TEORI

2.1 Computer Vision

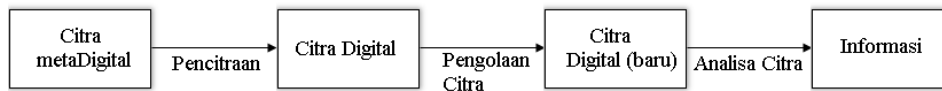
Computer Vision adalah suatu ilmu pada bidang komputer yang mempelajari bagaimana mesin mampu mengekstrak informasi dan dapat mengenali objek yang diamatinya (Irianto, Suhendro; Zaini, T. M.; Karmila, Sri. 2013). Cabang ilmu ini bersama *Artificial Intelligence* akan mampu menghasilkan *Visual Intelligence System*. Perbedaanannya adalah *Computer Vision* lebih mempelajari bagaimana komputer dapat mengenali objek yang diamatinya. *Computer Vision* adalah kombinasi dari:

- 1) Pengolahan citra : Bidang yang berhubungan dengan proses transformasi citra/gambar sehingga mendapatkan kualitas gambar yang lebih baik.
- 2) Pengenalan pola : Bidang yang berhubungan dengan proses identifikasi objek pada citra atau interpretasi citra yang bertujuan untuk mengekstrak informasi atau pesan yang disampaikan oleh citra.

2.2 Pengolahan Citra Digital

Citra atau *image* adalah angka, dari segi estetika, citra atau gambar adalah kumpulan warna yang bisa terlihat indah, memiliki pola, berbentuk abstrak dan lain sebagainya. Citra dapat berupa foto udara, penampang lintang (*cross section*) dari suatu benda, gambar wajah, hasil tomografi otak dan lain sebagainya. Dari segi ilmiah, citra adalah gambar 3-dimensi (3D) dari suatu fungsi, biasanya intensitas warna sebagai fungsi spatial x dan y . Di komputer, warna dapat dinyatakan, misalnya sebagai angka dalam bentuk skala *RGB*. Karena citra adalah angka, maka citra dapat diproses secara digital. Pengolahan citra digital (*Digital Image Processing*) adalah sebuah disiplin ilmu yang mempelajari tentang teknik-teknik mengolah citra. Citra yang dimaksud disini adalah gambar diam (foto) maupun gambar bergerak (yang berasal dari webcam). Sedangkan digital disini

mempunyai maksud bahwa pengolahan citra atau gambar dilakukan secara digital menggunakan komputer (Kusumanto, R. D.; Tompunu, Alan Novi, 2011, 1.1). Dalam bidang *computer vision*, secara umum proses yang terjadi seperti yang terlihat pada gambar 2.1.



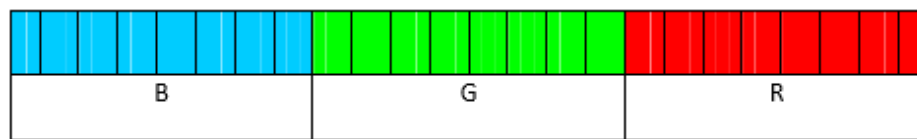
Gambar 2.1 Urutan pengolahan citra digital

Suatu citra digital melalui pengolahan citra digital (*digital image processing*) menghasilkan citra digital yang baru, termasuk didalamnya adalah perbaikan citra (*image restoration*) dan peningkatan kualitas citra (*image enhancement*). Sedangkan analisis citra digital (*digital image analysis*) menghasilkan suatu keputusan atau suatu data, termasuk didalamnya adalah pengenalan pola (*pattern recognition*). Dalam pengolahan citra ada beberapa tahapan yang dapat digunakan yaitu mengatur ukuran, menerapkan metode ekualisasi *histogram*, serta melalui tahapan segmentasi dengan menggunakan metode *k-mean clustering*.

2.3 Citra RGB

Ruang warna *RGB* adalah warna utama yang dimiliki citra dengan tiga warna utama yaitu *red*, *green*, dan *blue*. Rentang nilai yang dimiliki citra *RGB* dalam setiap piksel citra adalah 0 sampai dengan 225 (Riska, Suastika Yulia, 2015, 9.2: 18-26). Sebenarnya bagaimana citra disimpan dan dimanipulasi dalam komputer diturunkan dari teknologi televisi, yang pertama kali mengaplikasikannya untuk tampilan grafis komputer. Jika dilihat dengan kaca pembesar, tampilan monitor komputer akan terdiri dari sejumlah triplet titik warna merah (*RED*), hijau (*GREEN*) dan biru (*BLUE*). Tergantung pada pabrik monitornya untuk menentukan apakah titik tersebut merupakan titik bulat atau kotak kecil, tetapi akan selalu terdiri dari 3 *triplet red*, *green* dan *blue*. Citra dalam komputer tidak lebih dari sekumpulan sejumlah *triplet* dimana setiap *triplet* terdiri atas variasi tingkat

keterangan (*brightness*) dari elemen *red*, *green* dan *blue*. Representasinya dalam *citra*, *triplet* akan terdiri dari 3 angka yang mengatur intensitas dari *Red* (*R*), *Green* (*G*) dan *Blue* (*B*) dari suatu *triplet*. Setiap *triplet* akan merepresentasikan 1 *pixel* (*picture element*). Suatu *triplet* dengan nilai 67, 228 dan 180 berarti akan mengeset nilai R ke nilai 67, G ke nilai 228 dan B ke nilai 180. Angka-angka RGB ini yang seringkali disebut dengan *color values*. Pada format .bmp *citra* setiap *pixel* pada *citra* direpresentasikan dengan dengan 24 bit, 8 bit untuk R, 8 bit untuk G dan 8 bit untuk B, dengan pengaturan seperti pada gambar 2.2.



Gambar 2.2 Citra RGB

2.4 Histogram Equalization (HE)

Histogram Equalization merupakan suatu proses transformasi distribusi harga-harga intensitas *pixel* pada *citra* $I(x,y)$ menjadi distribusi harga intensitas *pixel* yang merata (*uniform*), sehingga memperbaiki kekontrasan *citra* secara keseluruhan. Proses ini bekerja dengan cara menyebarkan harga-harga intensitas *pixel* yang sering terjadi secara merata pada *citra*. *Citra* hasil normalisasi *Histogram Equalization* ditunjukkan pada gambar berikut :

$$i_{\text{new}} = \sum_{i=0}^{k-1} \frac{ni}{N} = \sum_{i=0}^{k-1} P(i)$$

Gambar 2.3 Normalisasi Citra

Kinerja tinggi *Histogram Equalization* dalam meningkatkan kontras gambar sebagai konsekuensi dari ekspansi rentang dinamis, Selain itu, *Histogram Equalization* juga meratakan *histogram*. Seperti yang telah dibahas diatas, *Histogram Equalization* dapat memperkenalkan perubahan yang signifikan dalam kecerahan gambar (Kaur, Manpreet; kaur, Jasdeep; kaur, Jappreet. 2011, 2.7: 137-141).

2.5 Citra *Grayscale*

Dalam komputasi, suatu citra digital *grayscale* atau *greyscale* adalah suatu citra dimana nilai dari setiap *pixel* merupakan sample tunggal. *Citra* yang ditampilkan dari citra jenis ini terdiri atas warna abu-abu, bervariasi pada warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. *Citra grayscale* berbeda dengan citra "hitam-putih", dimana pada konteks komputer, citra hitam putih hanya terdiri atas 2 warna saja yaitu "hitam" dan "putih" saja. Pada *citra grayscale* warna bervariasi antara hitam dan putih, tetapi variasi warna diantaranya sangat banyak. *Citra grayscale* seringkali merupakan perhitungan dari intensitas cahaya pada setiap *pixel* pada spektrum elektromagnetik *single band*. *Citra grayscale* disimpan dalam format 8 bit untuk setiap sample *pixel*, yang memungkinkan sebanyak 256 intensitas. Format ini sangat membantu dalam pemrograman karena manipulasi bit yang tidak terlalu banyak. Pada aplikasi lain seperti pada aplikasi *medical imaging* dan *remote sensing* biasa juga digunakan format 10,12 maupun 16 bit.



Gambar 2.4 citra grayscale

Ada beberapa metode untuk mengkonversi suatu citra warna *RGB* ke *grayscale*. Salah satu cara yang paling umum dipakai adalah rumus :

$Grayscale = 0.2989 * R + 0.5870 * G + 0.1140 * B$ (Tampubolon, Junius Karel; Hapsari, Widi; RE, Binarahandra. 2011)

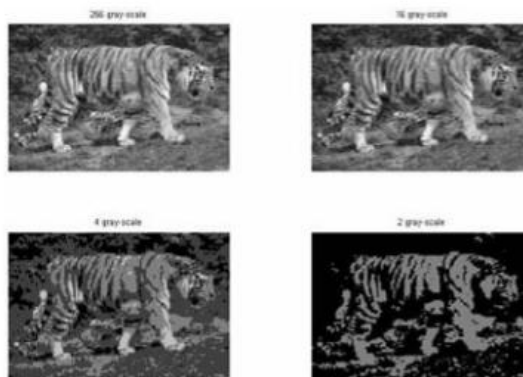
Kanal *Red* direpresentasikan dengan R, kanal *Green* dengan G dan kanal *Blue* dengan B.

2.6 Thresholding

Thresholding merupakan konversi citra hitam – putih ke *citra biner* dilakukan dengan cara mengelompokkan nilai derajat keabuan setiap *pixel* kedalam 2 kelas, hitam dan putih. Pada citra hitam putih terdapat 256 *level*, artinya mempunyai skala “0” sampai “255” atau [0,255], dalam hal ini nilai intensitas 0 menyatakan hitam, dan nilai intensitas 255 menyatakan putih, dan nilai antara 0 sampai 255 menyatakan warna keabuan yang terletak antara hitam dan putih. Dengan menggunakan *thresholding* maka derajat keabuan bisa diubah sesuai keinginan, misalkan diinginkan menggunakan derajat keabuan 16, maka tinggal membagi nilai derajat keabuan dengan 16. Proses *thresholding* ini pada dasarnya adalah proses pengubahan kuantisasi pada citra, sehingga untuk melakukan *thresholding* dengan derajat keabuan dapat digunakan rumus dimana : w = nilai sebelum *thresholding* x = nilai setelah *thresholding*

$$x = b.\text{int}\left(\frac{w}{b}\right)$$

Berikut ini contoh *thresholding* mulai di 256, 16, 4 dan 2.



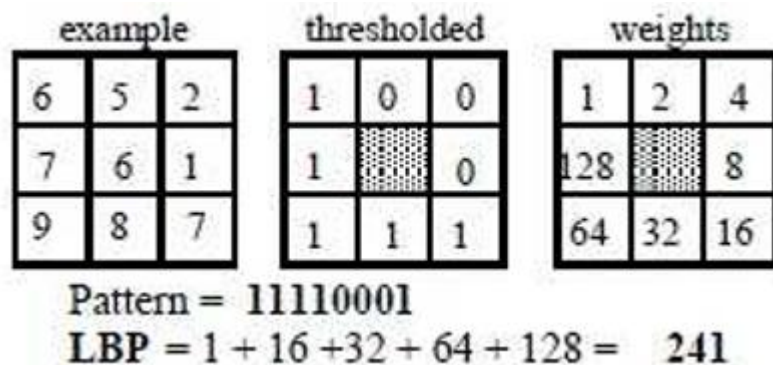
Gambar 2.5 thresholding

2.7 Local Binary Pattern (LBP)

Operator LBP adalah salah satu deskriptor tekstur terbaik dan telah banyak digunakan dalam berbagai aplikasi. *LBP* telah terbukti sangat diskriminatif dan keuntungan utamanya, yaitu variasi untuk perubahan tingkat abu-abu monoton

dan efisiensi komputasi, membuatnya cocok untuk tugas gambar menuntut analisis. Ide untuk menggunakan *LBP* untuk deskripsi wajah didukung oleh fakta wajah dapat dilihat sebagai komposisi pola mikro yang dapat dijelaskan oleh sebuah *operator* (Hasibuan, Muhammad Said; Suhendro, Irianto. 2012). *Local Binary Pattern (LBP)* didefinisikan sebagai ukuran tekstur *gray-scale invariant*, berasal dari definisi umum tekstur di daerah sekitar (Irianto, Suhendro; Zaini, T. M.; Karmila, Sri. 2013). Operator *LBP* dapat dilihat sebagai pendekatan kesatuan dengan model statistik dan struktur tradisional berbeda dari analisis tekstur. Secara sederhana, *LBP* adalah sebuah kode *biner* yang menggambarkan pola tekstur lokal. Hal ini dibangun dengan lingkungan batas dengan nilai abu-abu dari pusatnya.

Contoh komputasi *LBP* pada 3×3 *pixel* :



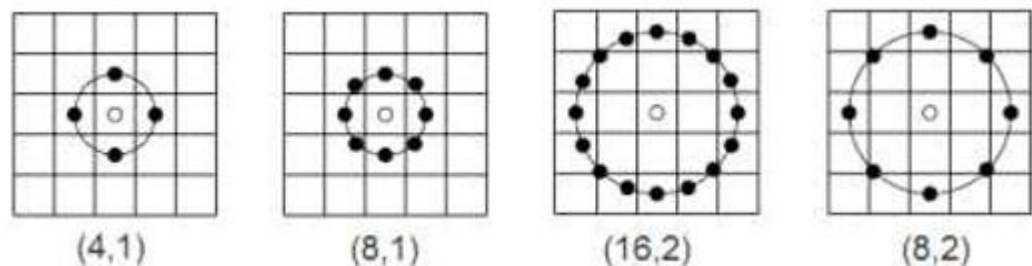
Gambar 2.6 Proses Perhitungan Pixel LBP

Setiap *pixel* memiliki nilai hasil *grayscale*, kemudian dilakukan *threshold* berpusat pada titik tengah. *Pixel* yang memiliki nilai sama atau lebih dibandingkan dengan titik tengah diberi nilai 1 selain itu diberi nilai 0. Kemudian nilai *LBP* didapat dari penjumlahan dua pangkat nilai angka yang bernilai satu.

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

Gambar 2.7 Rumus komputasi LBP

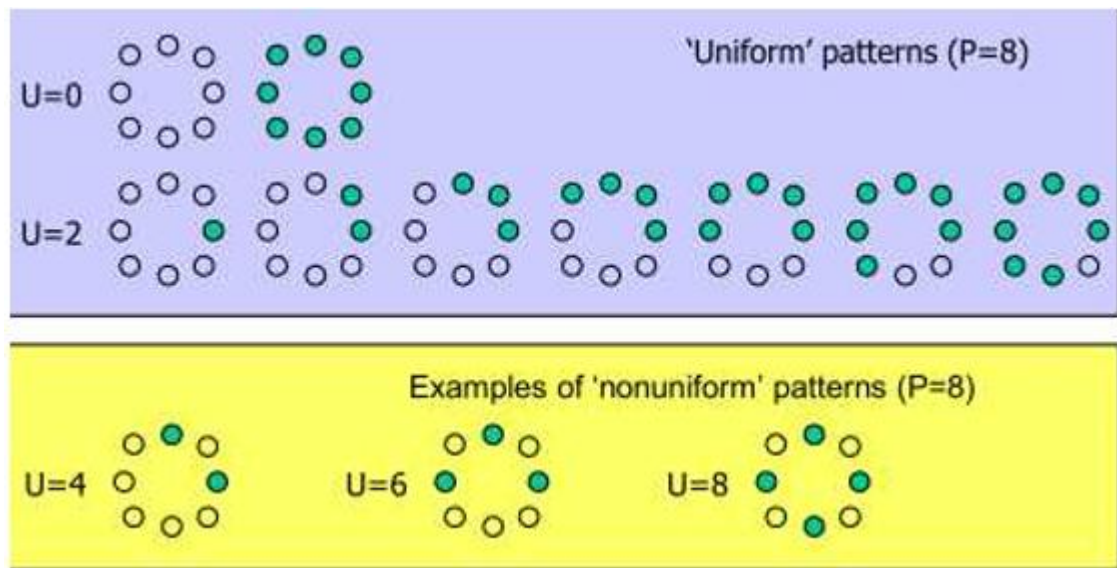
Operator pada *LBP* memiliki label yang ditandai dengan P dan R. P mewakili jumlah *pixel* tetangga yang digunakan dalam komputasi sementara R adalah radius antara *pixel* titik pusat dan *pixel* tetangga.



Gambar 2.8 Varian LBP

Dalam aplikasi analisis tekstur banyak diinginkan untuk memiliki fitur yang invarian atau kuat untuk rotasi gambar input. Sebagai *LBP*, pola P,R diperoleh dengan sirkuler sampel sekitar *pixel* pusat, rotasi gambar input memiliki dua efek: setiap lingkungan lokal diputar ke lokasi *pixel* lainnya, dan dalam masing-masing lingkungan, titik *sampling* pada lingkaran yang mengelilingi titik pusat diputar ke orientasi yang berbeda.

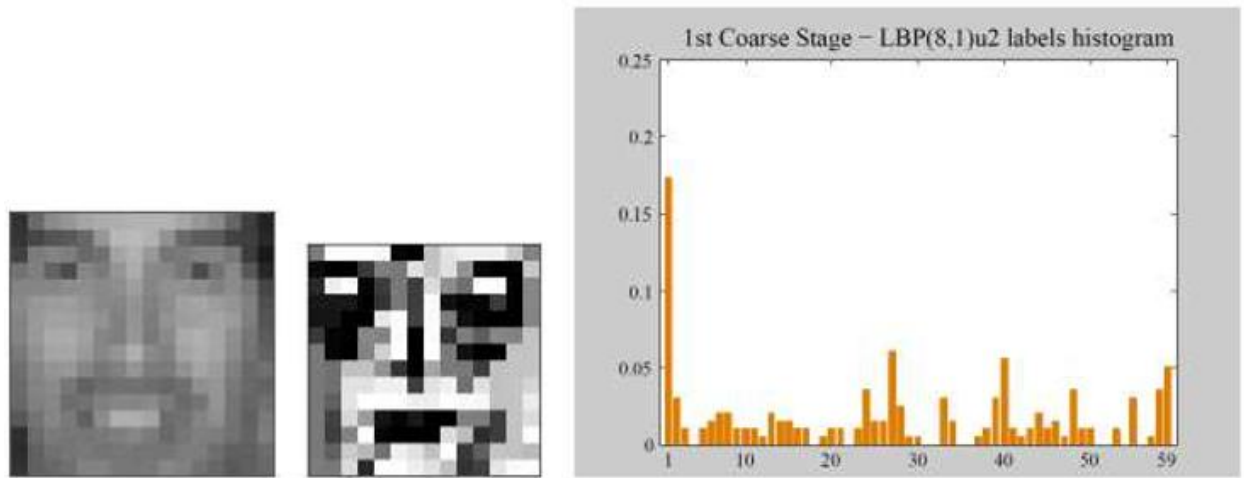
Perpanjangan pada operator aslinya menggunakan *Uniform Pattern*. Untuk ini, ukuran keseragaman pola yang digunakan : U ("pola") adalah jumlah *bitwise* transisi dari 0 ke 1 atau sebaliknya ketika pola *bit* dianggap melingkar. Pola *biner* lokal disebut seragam jika ukuran keseragaman adalah 2. Misalnya, pola 00000000 (0 transisi), 01110000 (2 transisi) dan 11001111 (2 transisi) adalah seragam sedangkan pola 11001001 (4 transisi) dan 01010011 (6 transisi) tidak. Dalam pemetaan *LBP* seragam ada yang terpisah *output label* untuk setiap pola seragam dan semua non-seragam pola ditugaskan ke label tunggal. Dengan demikian, jumlah label *output* yang berbeda untuk pemetaan pola-pola.



Gambar 2.9 Uniform Pattern

Pada proses *LBP*, dibentuk *histogram* dengan menambahkan nilai setiap blok sesuai dengan pola *biner* yang sama. Pola *biner* tersebut yang digunakan berdasarkan *pattern uniform* yang digunakan. Pada prakteknya suatu *image* dibagi menjadi beberapa bagian/blok. Hal ini dilakukan untuk mempercepat proses komputasi yang dilakukan.

Pada Gambar 2.10, proses ekstraksi wajah dilakukan dengan menggunakan *local binary pattern*. *Uniform pattern* yang digunakan adalah 2. Alasan digunakan pengambilan pola *biner* hanya pada $U=2$ karena pola *biner* $U=2$ mewakili sekitar 80% dari jumlah *pixel* yang ada (Lopez, 2013, p. 9). Sehingga hal ini tidak mengganggu proses identifikasi wajah. Dari 8 *biner* terdapat 256 pola *biner* yang di dapat dari 2 pangkat 8. Namun pola *biner uniform pattern* 2 hanya terdapat 58 pola *biner uniform*. Tapi perlu ditambah kan 1 lagi sebagai *non-uniform* sebagai perwakilan dari semua pola *non uniform*. Sehingga jumlah bin yang digunakan adalah 59 bin.



a) Contoh gambar wajah b) *LBP imagelabel* c) *Histogram LBP uniform pattern 2*

Gambar 2.10 Proses penentuan nilai pada LBP

2.8 OpenCV

OpenCV adalah suatu *library* gratis yang dikembangkan oleh *developer-developer Intel Corporation*. *Library* ini terdiri dari fungsi-fungsi *computer vision* dan *API (Application Programming Interface)* untuk *image processing high level* maupun *low level* dan sebagai optimasi aplikasi *realtime*. *OpenCV* sangat disarankan untuk *programmer* yang akan berkecukupan pada bidang *computer vision*, karena *library* ini mampu menciptakan aplikasi yang handal, kuat dibidang *digital vision*, dan mempunyai kemampuan yang mirip dengan cara pengolahan visual pada manusia, Karena *library* ini bersifat *cuma-cuma* dan sifatnya yang *open source*, maka dari itu *OpenCV* tidak dipesan khusus untuk pengguna arsitektur *Intel*, tetapi dapat dibangun pada hampir semua arsitektur. Saat ini para *developer* dari *Intel Corporation* telah membuat berbagai macam versi, yaitu: - *openCV* untuk bahasa pemrograman *C/C++*, - *openCV* untuk bahasa pemrograman *C#* (masih dalam tahap pengembangan), dan *openCV* untuk bahasa pemrograman *Java*. Untuk bahasa pemrograman *C#* dan *Java*, karena masih dalam tahap pengembangan, maka kita membutuhkan *library* lain sebagai pelengkap kekurangan yang ada.

Namun untuk bahasa pemrograman *C/C++* tidak memerlukan *library* lainnya untuk pemrosesan pada *computer vision*.

Berikut ini adalah fitur-fitur pada *library OpenCV*:

- Manipulasi data gambar (alokasi memori, melepaskan memori, kopi gambar, setting serta konversi gambar)
- *Image/Video I/O* (Bisa menggunakan camera yang sudah didukung oleh *library* ini)
- Manipulasi matriks dan vektor serta terdapat juga *routines linear algebra* (*products, solvers, eigenvalues, SVD*)
- *Image processing* dasar (*filtering, edge detection*, pendeteksian tepi, *sampling* dan interpolasi, konversi warna, operasi morfologi, *histograms, image pyramids*)
- Analisis struktural
- Kalibrasi kamera
- Pendeteksian gerak
- Pengenalan objek
- Basic *GUI* (Display gambar/*video*, mouse/*keyboard kontrol, scrollbar*)
- *Image Labelling* (*line, conic, polygon, text drawing*)

2.9 Flowchart Diagram

(Indrajani, 2011) mengemukakan bahwa *flowchart* merupakan penggambaran secara grafik dari langkah-langkah dan urutan prosedur suatu program. Biasanya mempermudah penyelesaian masalah yang khususnya perlu dipelajari dan dievaluasi lebih lanjut.

(Fairuz El Sahid, 2010) mengemukakan bahwa *Flowchart* adalah bagan (*chart*) yang menunjukkan alir (*flow*) di dalam program atau prosedur sistem secara

logika. Bagan alir digunakan terutama untuk alat bantu komunikasi dan untuk dokumentasi.

Flowchart di bedakan menjadi 5 jenis *flowchart*, antara lain *system flowchart*, *document flowchart*, *schematic flowchart*, *program flowchart*, *process flowchart*. Masing-masing jenis *flowchart* akan di jelaskan berikut ini.

1. System Flowchart

System flowchart dapat didefinisikan sebagai bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. Bagan ini menjelaskan urutan dari prosedur-prosedur yang ada di dalam sistem. Bagan alir sistem menunjukkan apa yang dikerjakan di sistem.

2. Document Flowchart

Bagan alir dokumen (*document flowchart*) atau disebut juga bagan alir formulir (*form flowchart*) atau *paperwork flowchart* merupakan bagan alir yang menunjukkan arus dari laporan dan formulir termasuk tembusan-tembusannya.

3. Schematic Flowchart

Bagan alir skematik (*schematic flowchart*) merupakan bagan alir yang mirip dengan bagan alir sistem, yaitu untuk menggambarkan prosedur di dalam sistem. Perbedaannya adalah, bagan alir skematik selain menggunakan simbol-simbol bagan alir sistem, juga menggunakan gambar-gambar komputer dan peralatan lainnya yang digunakan. Maksud penggunaan gambar-gambar ini adalah untuk memudahkan komunikasi kepada orang yang kurang paham dengan simbol-simbol bagan alir. Penggunaan gambar-gambar ini memudahkan untuk dipahami, tetapi sulit dan lama menggambarinya.

4. Program Flowchart





Bagan alir program (*program flowchart*) merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program. Bagan alir program dibuat dari




derivikasi bagan alir sistem. Bagan alir program dapat terdiri dari dua macam, yaitu bagan alir logika program (program *logic flowchart*) dan bagan alir program komputer terinci (*detailed computer program flowchart*). Bagan alir logika program digunakan untuk menggambarkan tiap-tiap langkah di dalam program komputer secara logika. Bagan alat- logika program ini dipersiapkan oleh analis sistem. Gambar berikut menunjukkan bagan alir logika program. Bagan alir program komputer terinci (*detailed computer program flow-chart*) digunakan untuk menggambarkan instruksi-instruksi program komputer secara terinci. Bagan alir ini dipersiapkan oleh pemrogram.







5. Process Flowchart



Bagan alir proses (*process flowchart*) merupakan bagan alir yang banyak digunakan di teknik industri. Bagan alir ini juga berguna bagi analis sistem untuk menggambarkan proses dalam suatu prosedur.

Berikut ini merupakan notasi atau symbol-simbol dalam penggambaran *flowchart* yang di kutip dari artikel elektronik Fairuz El Sahid:

Simbol	Keterangan
	<i>Symbol Off-line Connector</i> (Simbol untuk keluar/masuk prosedur atau proses dalam lembar/halaman yang lain)
	<i>Symbol Connector</i> (Simbol untuk keluar/masuk prosedur atau proses dalam lembar/halaman yang sama)
	<i>Symbol Process</i> (Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer)
	<i>Symbol Manual Operation</i> (Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh komputer)

	<i>Symbol Decision</i> (Simbol untuk kondisi yang akan menghasilkan beberapa kemungkinan jawaban/aksi)
	<i>Symbol Predefined Process</i> (Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage)
	<i>Symbol Terminal</i> (Simbol untuk permulaan atau akhir dari suatu program)

	<i>Symbol Off-line Storage</i> (Simbol yang menunjukkan bahwa data di dalam simbol ini akan disimpan)
	<i>Symbol Manual Input</i> (Simbol untuk pemasukan data secara manual on-line keyboard)
	<i>Symbol input-output</i> (Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya)
	<i>Symbol magnetic-tape unit</i> (Simbol yang menyatakan input berasal pita magnetic atau output disimpan ke pita <i>magnetic</i>)
	<i>Symbol punched card</i> (Simbol yang menyatakan <i>input</i> berasal dari kartu atau <i>output</i> ditulis ke kartu)
	<i>Symbol disk and on-line storage</i> (Simbol untuk menyatakan <i>input</i> berasal dari <i>disk</i> atau <i>output</i> disimpan ke <i>disk</i>)

	<p><i>Symbol display</i> (Simbol yang menyatakan peralatan <i>output</i> yang digunakan yaitu layar, <i>plotter</i>, <i>printer</i>, dan sebagainya)</p>
	<p><i>Symbol dokumen</i> (Simbol yang menyatakan <i>input</i> berasal dari dokumen dalam bentuk kertas atau <i>output</i> dicetak ke kertas)</p>

2.10 Structured Query Language (SQL)

Structured Query Language (SQL) merupakan bahasa *query* standar yang digunakan untuk mengubah dan merancang hubungan yang digunakan untuk mengubah *input* menjadi *output* yang dibutuhkan (Eric L.Brown, 2009, p2). Menurut (Connolly, Begg, 2010, p184) *SQL* adalah suatu bahasa *database* yang dapat melaksanakan tugas secara minimal dari perintah struktur sintaks yang harus relevan dan *portable* yang sesuai dengan standar yang telah ditentukan. Jadi dari pengertian diatas dapat disimpulkan bahwa *SQL* merupakan sebuah bahasa yang dipergunakan untuk mengakses data dalam sebuah *database* dengan menggunakan bahasa standar yang telah ditentukan .

2.10.1 Keunggulan SQL

SQL mempunyai keunggulan diantaranya yaitu :

1. *Performance* memiliki kecepatan yang dapat menangani *query* secara cepat tanpa harus melewati proses yang rumit
2. *Low Cost* menyediakan *open source* yang berlisensi secara gratis dalam bentuk *commercial license*.
3. *Easy To Use* menyediakan sebagian besar *database* yang menggunakan sintaks *SQL*. Dengan kemudahan dalam proses *set up* dibanding produk-produk yang serupa.

4. *Portability* dapat berisikan secara *stability* pada berbagai OS seperti *Windows, Linux, Mac os*, dan lain-lain
5. *Source Code* memudahkan pengguna untuk mengontrol dan memodifikasi *source code SQL*