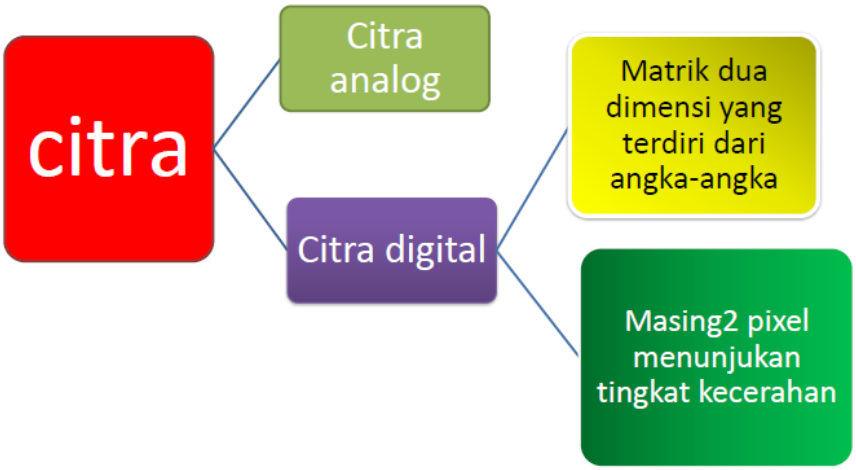
BAB 2
LANDASAN TEORI

2.1 Citra

Pengertian citra atau *image* merupakan istilah lain dari gambar, yang merupakan informasi berbentuk visual. Suatu citra adalah fungsi intensitas 2 dimensi f(x,y), dimana x dan y adalah koordinat spasial dan f pada titik (x,y) merupakan tingkat kecerahan (*brightness*) suatu citra pada suatu titik[3].



Gambar 2. 1 Representasi Citra

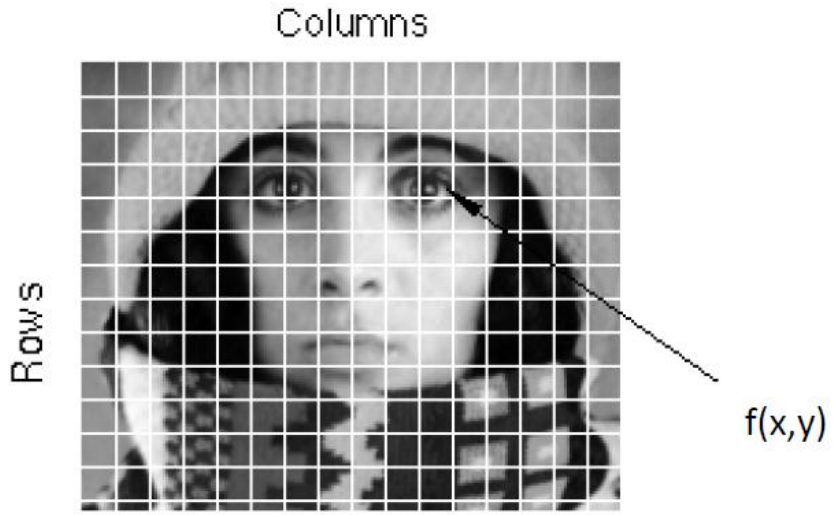
Citra ada 2 macam :

1. Citra Kontinu yaitu, citra yang dihasilkan dari sistem optik yang menerima sinyal analog, misalnya, mata manusia dan kamera analog.
2. Citra Diskrit yaitu, citra yang dihasilkan melalui proses digitalisasi terhadap citra kontinu.

Beberapa sistem optik dilengkapi dengan fungsi digitalisasi sehingga mampu menghasilkan citra diskrit, misalnya kamera digital dan scanner.

2.2 Citra Digital

Citra digital adalah citra *f(x,y)* yang telah dilakukan digitalisasi baik koordinat area maupun *brightness* level. Nilai f di koordinat *(x,y)* merupakan *brightness* atau *grayness* level dari citra pada titik tersebut[1]. Citra yang dihasilkan direkam datanya bersifat kontinu harus diubah dahulu menjadi citra digital dengan konversi sehingga dikenali komputer. Karena komputer hanya dapat mengolah data digital dan citra harus mempunyai format tertentu. Proses tersebut disebut digitasi, yaitu membuat kisi-kisi arah *horizontal* dan *vertical* sehingga terbentuk array 2 dimensi.



Gambar 2. 2 Representasi Citra Dua Dimensi

Representasi dari sebuah citra dua dimensi sebagai sebuah kumpulan nilai digital yang disebut elemen gambar atau piksel.*Pixel* adalah satuan terkecil dari citra yang mengandung nilai terkuantisasi yang mewakili *brightness* dari sebuah warna pada sebuah titik tertentu.Citra digital merupakan suatu matriks dimana indeks baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya (yang disebut sebagai elemen gambar atau piksel) menyatakan tingkat keabuan pada titik tersebut.

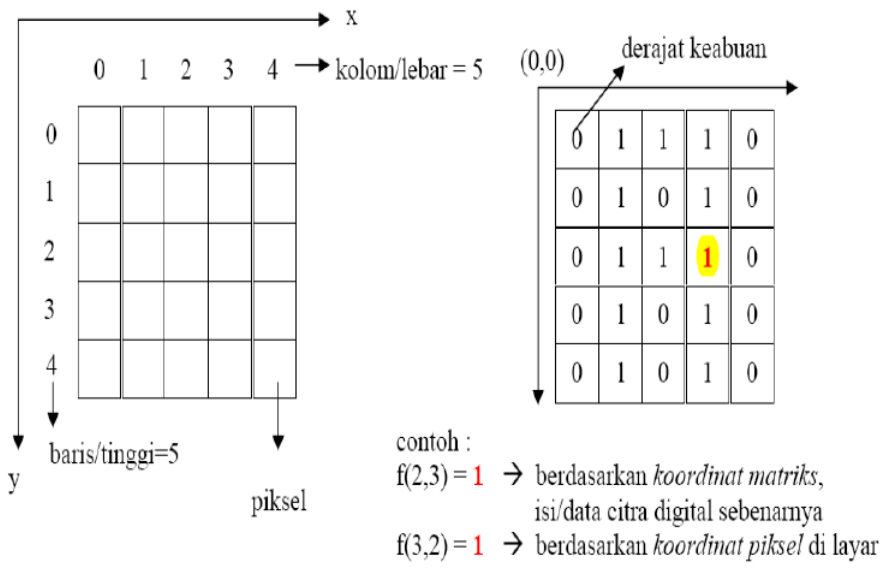
Citra digital berukuran N x M (tinggi = N, lebar = M) dinyatakan dengan matriks N x M.

N = jumlah baris $0 \leq y \leq N-1$

M = jumlah baris $0 \leq x \leq M-1$

Citra digital dipresentasikan sebagai matriks berukuran N x M dimana N = baris dan M = kolom sebagai berikut :

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



Gambar 2. 3 Ilustrasi Matriks Citra Digital

Masing-masing elemen pada citra digital (berarti elemen matriks) disebut *image element* atau piksel. Jadi, citra yang berukuran N x M mempunyai NM buah piksel.

Proses digitalisasi koordinat (x,y) dikenal sebagai pencuplikan citra *(image sampling),* sedangkan proses digitalisasi derajat keabuan f(x,y) disebut kuantisasi derajat keabuan *(gray-level quantication).*

2.2.1 Elemen – elemen Citra Digital

Citra digital mengandung sejumlah elemen-elemen dasar. Elemen-elemen dasar inilah yang dimanipulasi dalam pengolahan citra. Elemen-elemen yang penting diantaranya adalah sebagai berikut :

2.2.1.1 Kecerahan (*brightness*)

Kecerahan disebut juga sebagai intensitas cahaya yang dipancarkan pixel dari citra yang dapat ditangkap pleh sistem penglihatan. Kecerahan pada suatu titik (piksel) di dalam suatu citra sebenarnya adalah intensitas rata-rata dari suatu area yang melingkupinya.

2.2.1.2 Kontras *(contrast)*

Kontras menyatakan seberapa terang (*lightness*) dan gelap (*darkness*) dalam suatu citra. Citra dengan kontras rendah dicirikan oleh sebagian besar komposisi citranya adalah terang atau sebagian besar gelap. Citra dengan kontras yang baik, komposisi gelap dan tersebar secara merata.

2.2.1.3 Kontur *(contour)*

Kobtur adalah keadaan yang ditimbulkan oleh perubahan intensitas pada *pixel-pixel* yang bertetangga. Karena adanya perubahan intensitas inilah, maka tepi-tepi (*edge*) objek pada citra dapat dideteksi.

2.2.1.4 Warna *(color)*

Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek. Setiap warna mempunyai panjang gelombang ( ) yang berbeda-beda. Warna yang diterima oleh sistem visual (mata) merupakan hasil kombinasi cahaya dengan panjang gelombang yang berbeda-beda. Kombinasi warna yang memberikan rentang warna yang paling lebar adalah *red (R), green (g), blue (B).*

2.2.1.5 Bentuk *(shape)*

Bentuk adalah properti intrinstik dari objek tiga dimensi. Bentuk merupakan properti intrinsik utama untuk sistem visual manusia karena lebih sering menginterpretasikan suatu objek berdasarkan bentuknya daripada elemen lainya.

2.2.1.6 Tekstur *(texture)*

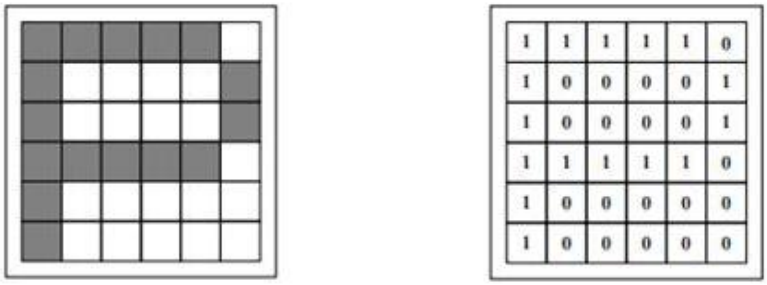
Tekstur dicirikan sebagai distribusi spasial dari derajat keabuan di dalam sekumpulan *pixel-pixel,* Tekstur merupakan karakteristik untuk menganalisis permukaan berbagai jernis citra objek.

2.2.2 Jenis Citra Digital

Berdasarkan format penyimpanan nilai warnanya Citra Digital terbagi menjadi tiga macam[2] yaitu :

2.2.2.1 Citra Biner (Monokrom)

Citra yang hanya memiliki dua nilai yaitu 1 dan 0. Citra biner juga disebut sebagai citra B&W *(black and white)* atau citra monokrom, setiap *pixel* pada citra biner cukup direpresentasikan dengan 1 bit.



Gambar 2. 4 Citra Biner dan Array Citra Biner

2.2.2.2 Citra *Greyscale* (skala keabuan)

Pada citra dengan tipe *grayscale*, setiap piksel mewakili derajat keabuan dengan nilai antara 0 (hitam) samapai 255 (putih). Nilai intensitas paling rendah merepresentasikan warna hitam dan nilai intensitas paling tinggi merepresentasikan warna putih. Pada jangkauan nilai 0 sampai 255, ini berarti bahwa piksel memiliki ukuran 8 bit atau 1 byte.



Gambar 2. 5 Deret warna *grayscale*



Gambar 2. 6 Citra *Grayscale*

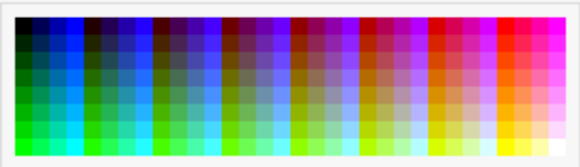
Citra grayscale merupakan citra satu kanal, dimana citra f(x,y) merupakan fungsi tingkat keabuan dari hitam keputih, x menyatakan variabel kolom atau posisi piksel di garis jelajah dan y menyatakan variabel kolom posisi piksel di garis jelajah. Intensitas f dari gambar hitam putih pada titik (x,y) disebut derajat keabuan (gray level), yang dalam hal ini derajat keabuannya bergerak dari hitam keputih. Derajat keabuan memiliki rentang nilai dari Imin sampai Imax, atau Imin < f < Imax, selang (Imin, Imax) disebut skala keabuan.

2.2.2.3 Citra Warna (8 bit)

Setiap *pixel* dari citra warna (8 bit) hanya di wakili oleh 8 bit jumlah warna maksimum yang dapat digunakan adalah 256 warna. Ada dua jenis citra warna 8 bit. Pertama, citra warna 8 bit dengan menggunakan palet warna 256 dengan setiap paletnya memiliki pemetaan nilai (*colormap*) RGB tertentu. Model ini lebih sering digunakan. Kedua, setiap piksel memiliki format 8 bit sebagai berikut,

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
R	R	R	G	G	G	B	B

Bentuk kedua ini dinamakan 8 bit *truecolor*. Berikut ini adalah warna-warnanya



Gambar 2. 7 Susunan Warna 8bit *Truecolor*



Gambar 2. 8 Citra 8-bit dengan palet dan citra 8-bit color

Secara umum, warna yang ada 8bit color *image* didapatkan dari *palette*yang terdiri dari 256 (8 x 8 x 4) warna, dimana tiap entri pada palet diberikan RGB tersendiri yang unik.

2.2.2.4 Citra (16 bit)

Citra warna 16 bit (biasanya disebut sebagai citra *high color*) dengan setiap pikselnya diwakili dengan 2 byte memory (16 bit). Warna 16 bit memiliki 65.635 warna. Dalam formasi bitnya, nilai merah dan biru mengambil tempat di 5 bit di kanan dan kiri. Komponen hijau memiliki 5 bit ditambah 1 bit eksra. Pemilihan komponen hijau dengan deret 6 bit dikarenakan penglihatan manusia lebih sensitif terhadap warna hijau.

Bit -15	Bit -14	Bit -13	Bit -12	Bit -11	Bit -10	Bit -9	Bit -8	Bit -7	Bit -6	Bit -5	Bit -4	Bit -3	Bit -2	Bit -1	Bit -0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B

Berikut ini adalah deret warna yang dihasilkan dari warna 16 bit.



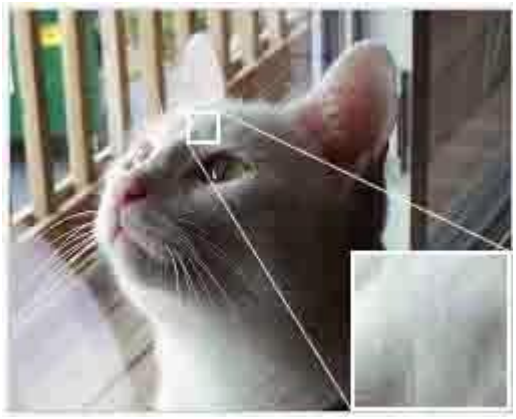
Gambar 2. 9 Deret warna 16-bit



Gambar 2. 10 Citra warna 16-bit

2.2.2.5 Citra (24 bit)

Setiap *pixel* dari citra warna 24 bit diwakili sehingga total 16.777.216 variasi warna. Variasi ini sudah lebih dari cukup untuk memvisualisasikan seluruh warna yang dapat dilihat penglihatan manusia. Penglihatan manusia dipercaya hanya dapat membedakan hingga 10 juta warna saja. Setiap poin informasi *pixel* (RGB) disimpan ke dalam 1 byte data. 8 bit pertama menyimpan nilai baru, kemudian diikuti dengan hijau pada 8 bit kedua dan pada 8 bit terakhir merupakan warna merah.



Gambar 2. 11 Citra warna 24-bit

Dua puluh empat bit data tiap piksel untuk warna menyebabkan warna yang tercipta akan sangat detail dan dapat mencakup seluruh warna yang di-'terjemahkan' oleh mata manusia normal, sehingga *image* jenis inicocok dimanfaatkan untuk keperluan yang memerlukan *advance graphics* sebagai gambarnya. Dalam sistem *macintosh*, 24-bit color berarti "sejuta warna".

2.2.3 Karakteristik Citra Digital

Setiap citra digital memiliki beberapa karakteristik, antara lain yaitu :

2.2.3.1 Ukuran Citra Digital

Ukuran citra (*image size*) menyatakan ukuran banyaknya *pixel*penyusun citra raster yang dinyatakan dalam matrik 2 dimensi, yaitu (X x Y) piksel, dimana X menyatakan ukuran banyaknya piksel perbaris pada arah horizontal sedangkan Y menyatakan ukuran banyaknya *pixel* perkolom pada arah vertikal. Setiap contoh, citra digital berukuran 800 x 600 *pixel*, terdiri dari 800 x 600 piksel = 480.000 piksel, dengan susunan 800 piksel setiap baris pada arah horizontal dan 600 piksel setiap kolom pada arah vertikal.

2.2.3.2 Resolusi

Resolusi citra merupakan tingkat detail suau citra. Semakain tinggi resolusi citra maka akan semakin tinggi pula tingkat detail dari citra tersebut. Resolusi spasial menghasilkan jumlah *pixel* per-satuan panjang. Resolusi spasial dari sebuah monitor komputer adalah hingga 72 hingga 100 garis per-inchi atau dalam resolusi 72 hingga 100 ppi. Di dalam penginderaan jauh ada 4 istilah resolusi yaitu :

a. Resolusi Spasial

   Ialah ukuran terkecil objek yang dapat direkam oleh sebuah sistem sensor. Dengan kata lain maka resolusi spasial mencerminkan kerincian informasi informasi yang dapaet disajikan oleh suatu sistem sensor.

b. Resolusi Spektrum

   Membedakan intensitas ke dalam beberapa spektrum. Citra multi spektrum akan memberikan spektrum atau panjang gelombang yang lebih baik yang akan digunakan untuk menampilkan warna.

c. Resolusi Temporal

   Ialah frekuensi perekaman ulang daerah yang sama. Resolusi temporal berkaitan dengan video, suatu video merupakan kumpulan frame statis yang mberupa citra berurutan dan tampilan secara cepat. Resolusi temporal memberikan jumlah frame yang dapat ditampilkan disetiap detik dengan satuan *frame per second* (fps).

d. Resolusi Radiomatik

   Ialah kepekaan sensor terhadap perbedaan terkecil kekuatan sinyal. Dengan sensor terminal misalnya, kalau sensor 1 mampu merekam beda suhu terkecil0,2 C dan sensor 2 mampu merekam beda suhu terkecil 0,5 C; berarti resolusi radiomatik sensor 1 lebih baik dari pada sensor 2.

2.2.4 Format Citra Digital

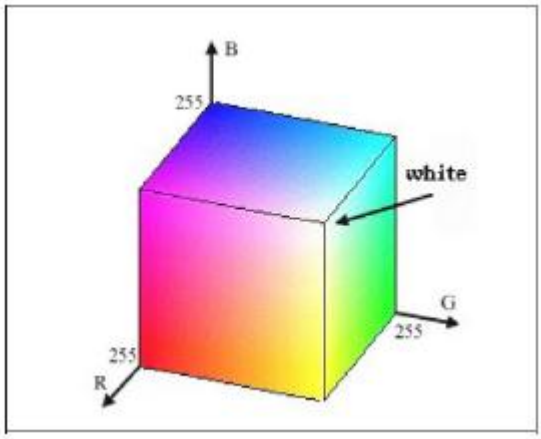
Citra digital dapat disimpan dalam berbagai macam format, format file menentukan bagaimana informasi data dipresentasikan dalam suatu file. Tiap format file memiliki kelebihan dan kelemahan pada masing-masing format tersebut dalam sistem operasi Windows biasanya format file dapat dibedakan dari namanya saja yaitu diakhiri dengan titik dan diikuti dengan tiga atau empat huruf terakhir. Beberapa format file pada citra digital[1] antara lain :

2.2.4.1 BMP(*Bitmap*)

Kriteria yang paling penting dari citra ini adalah kedalaman warna yaitu berapa banyak bit per *pixel* yang didefinisikan dari sebuah warna (Renaldi Munir, 2005).

Bitmap dengan mengikuti kriteria tadi maka dapat dilihat :

a. 8 bit = 256 warna (256 *gray scales*).
b. 24 bit = 16.777.216 warna



Gambar 2. 12 Warna bitmap

Secara umum dapat dikatakan semakin banyaknya warna, maka akan diperlukan keamanan yang tinggi dikarenakan bitmap memiliki area yang sangat luas dalam sebuah warna yang seharusnya dihindari. Dilihat dari kedalaman atau kejelasan dari sebuah warna, bitmap dapat mengambil sejumlah data tersembunyi

dengan perbandingan sebagai berikut (ukuran ratio dari bitmap dalam byte = ukuran dari data yang disembunyikan) :

a. 8 bit = 256 warna : 8 : 1
b. 24 bit = 16.777.216 : 8 : 1

Perbandingan tersebut diperoleh dari penentuan LSB dalam suatu byte, ukuran citra 8 bit letak LSB adalah pada bit terakhir sedangkan untuk citra 24 bit letak LSB adalah pada bit ke-8, bit ke-16 dan ke-24 dimana masing-masing byte memiliki warna merah (red) , warna hijau (green) dan warna biru (blue).

2.2.4.2 GIF (*Graphic Interchage Format*)

GIF menggunakan maksimal 8 bit warna (2^8 = 256 warna) yang dibuat oleh *Compuserver* pada tahun 1987 untuk menyimpan berbagai gambar dengan format *bitmap* menjadi file yang mudah untuk diubah pada jaringan komputer. *GIF* adalah file format *graphic* yang paling tua pada Web, dan begitu dekatnya file format ini dengan web pada saat itu sehingga para *Browser* meggunakan format ini.

File GIF dapat disimpan dalam dua jalan yaitu secara berurutan (dari atas ke bawah) dan pembagian dengan baris (8 baris, 4 baris dan 2 baris), Pembagian baris pada gambar dengan resolusi gambar yang rendah dengan cepat dimana secara gradual datangnya untuk menjadi lebih fokus, dengan *expense* dari penambahan kapasitas file.

Terdapat dus tipe dari GIFs, antara lain :

1. **GIF87a:** *support* dengan *interlancing* dan kapasitas dari beberapa file. Teknik itu dinamakan GIF87 karena pada tahun 1987 standar ini ditemukan dan dijadikan standar.
2. **GIF89a:** adalah kelanjutan dari spesifikasi GIF87a dan penambahan pada *transparency,* pemberian tulisan dan animasi daritext dan grafik.

2.2.4.3 JPEG (*Joint Photographic Experts Group*)

JPEG Menggunakan 24 bit warna (2^{24} = 16 juta warna) dan dirancang untuk kompresi beberapa *fill-color* atau *gray-scale* dari suatu gambar yang asli, seperti pemandangan asli di dunia ini. JPEG bekerja dengan baik pada *continous tone images* seperti *photographs* tetapi tidak terlalu bagus pada ketajaman gambar dan seni pewarnaaan seperti penulisan, kartun yang sederhana atau gambar yang menggunakan banyak garis. JPEG sudah mendukung untuk 24-bit *color depth* atau sama dengan 16,7 juta warna (224 = 16.777.216 warna), *progressive JPEGs (p-JPEGs)* adalah tipe dari beberapa persen lebih kecil dibandingkan *baseline* JPEGs.. Tetapi keuntungan dari JPEG dan tipe-tipenya terlihat pada langkanh-langkahnya sama seperti *interlaced* GIFs.

JPEG dirancang untuk mengeksploitas tingkat dari mata kita yakni bahwa mata kita tidak akan dapat membedakan perubahan yang lambat terang dan warna dibandingan dengan perbedaan suatu jarak apakah jauh atau dekat. Untuk itu JPEG sangat baik digunakan pada fotografi dan monitor 80-bit.

2.2.4.4 PNG (Portable Network Graphics)

Format PNG digunakan pada citra *grayscale* citra dengan palet warna, dan juga citra *full-color*. Format ini ini dibagi mejadi 2 tipe :

a. **PNG-8 :** menggunakan 8-bit warna, kurang kompatibel (tidak didukung) oleh browser, tetap biasanya mempunyai hasil kompresi yang lebih kecil dari format GIF. Berbeda dengan GIF yang telah dipatenkan. Format PNG bersifat bebas paten.
b. **PNG-**24 **:**menggunakan 24-bit warna, hampir sama dengan PNG-8 tetapi mempunyai ukuran yang lebih besar warna yang lebih banyak.

2.3 *Computer Vision*

Ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali objek yang diamati atau diobservasi. Ada hakikatnya komputer vision mencoba meniru cara kerja sistem visual manusia (*human vision*). Sebuah komputer yang menyerupai kemampuan manusia dalam menangkap sinyal visual (*human sight*) dilakukan dalam empat tahapan proses dasar yaitu proses penangkapan citra (*image acquisition*), proses pengolahan citra (*image processing*), analisa data citra (*image analysis*) dan proses pemahaman data citra (*image understanding*).

Proses dalam komputer vision meliputi[13] :

a. *Image Acqusition*

   Image Acqusition pada manusia dimulai dengan mata, kemudian informasi visual diterjemahkan ke dalam suatu format yang kemudian dapat dimanipulasi oleh otak. Senada dengan proses di atas, komputer vision membutuhkan sebuah mata untuk menangkap sebuah sinyal visual.

b. *Image Processing*

   Tahapan kedua adalah komputer vision akan melibatkan sejumlah manipulasi utama (*initial manipulation*) dari data biner tersebut. Image processing membantu peningkatan dan perbaikan kualitas image, sehingga dapat dianalisis dan di olah lebih jauh secara lebih efisien.

c. *Image Analysis*

   *Eksplorasi scene*ke dalam bentuk karakteristik utama dari objek melalui suatu proses investigasi merupakan tahapan image analisis. Sebuah program komputer akan mulai melihat melalui bilangan biner yang mepresesntasikan informasi visual untuk mengidentifikasikan fitur-fitur spesifikasi dan karakteristiknya.

d. *Image Understanding*

   Tahapan terakhir dari komputer vision adalah bagaimana komputer bisa memiliki pengetahuan tentang sebuah objek yang ada dalam scene.

2.3.1 Komputer Grafik

Computer Graphics dapat diartikan sebagai perangkat alat yang terdiri dari hardware dan software untuk membuat gambar, grafik atau citra realistic untuk seni, game, komputer, image, dan film animasi.

Komputer grafik merupakan bagian yang paling menarik dari bidang imlu komputer, ini dikarenakan untuk dapatmemahaminya dengan baik diperlukan kemampuan matematika dan juga keahlian untuk memprogram dan kreatifitas.

2.4 *Image Processing*

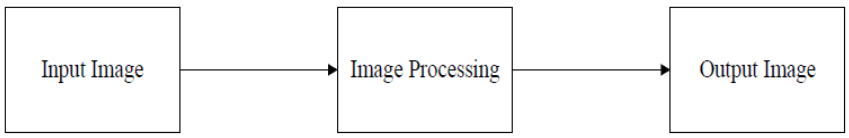
Pengertian sederhana dari image processing adalah manipulasi dan analisis suatu informasi gambar oleh komputer. Sedangkan yang dimaksud dengan informasi gambar di sini adalah gambar visual dalam dua dimensi. Segala operasi untuk memperbaiki, mendeteksi, menganalisis, atau mengubah suatu gambar disebut image processing[13].

Konsep dasar dari sistem dari image processing diambil dari kemampuan indera penglihatan manusia yang selanjutnya dihubungkan dengan kemampuan otak manusia. Dalam sejarahnya, image processing telah diaplikasikan dalam berbagai bentuk, dengan tingkat kesuksesan yang cukup besar. Seperti berbagai cabang ilmu lainnya, image processing menyangkut pula berbagai gabungan cabang-cabang ilmu, seperti optik, elektronik, matematika, fotografi, dan teknologi komputer. Pada umumnya tujuan dari image processing adalah mentransformasikan atau ada banyak cara yang dapat diaplikasikan dalam suatu operasi image processing, yang sebagian besar dalam bentuk optikal. menganalisis suatu gambar sehingga informasi baru tentang gambar dibuat lebih jelas.

Berbagai bidang telah banyak menggunakan aplikasi dari image processing baik dibidang komersial, industri, dan medis. Bahkan bidang militer telah menggunakan perkembangan dunia digital image processing ini. Pada

umumnya tujuan dari image processing adalah mentransformasikan atau menganalisis suatu gambar sehingga informasi baru tentang gambar dibuat lebih jelas. Ada banyak cara yang dapat diaplikasikan dalam suatu operasi image processing.

Berikut ini adalah bagian proses dari *image processing.*



Gambar 2. 13 Fase *Image Processing*

2.4.1 Pengenalan Pola

Pengenalan pola dapat dikatakan sebagai kemampuan mengenali objek berdasarkan ciri-ciri dan pengetahuan yang pernah diamatinya dari objek-objek tersebut. Tujuan dari pengenalan pola adalah mengklasifikasi dan mendeskripsikan pola atau objek kompleks melalui pengetahuan sifat-sifat atau ciri-ciri objek tersebut. Ada tiga pendekatan dalam pengenalan pola yaitu secara sintaks, statistik, dan semantik. Pengenalan pola secara sintaks dilakukan berdasarkan ciri-ciri objek. Pengenalan pola secara statistik dilakukan berdasarkan komputasi matematis. Pendekatan dengan semantik berarti pola dikenali dalam tataran yang lebih abstrak.

2.5 *Grayscaling*

Citra digital dapat diwakili oleh format warna RGB (*Red-Green-Blue*) untuk setiap titiknya, dimana setiap komponen warna memiliki batasan sebesar 1 byte. Jadi untuk masing-masing komponen R, G, dan B mempunyai variasi dari 0 sampai 255. Total variasi yang dihasilkan untuk sistem dengan format warna RGB adalah 256 x 256 x 256 atau 16.777.216 jenis warna. Karena setiap komponen

warna memiliki batasan sebesar 1 byte atau 8 bit, maka total untuk mempresentasikan warna RGB adalah 8+8+8 = 24 bit.

Gray-level adalah tingkat adalah tingkat warna abu-abu dari pixel, dapat juga dikatakan tingkat cahaya dari sebuah pixel (Prasetyo, 2011). Maksudnya nilai terkandung dalam pixel menunjukan tingkat terangnya pixel tersebut dari hitam ke putih. Biasanya diterapkan nilainya antara 0 hingga 255 (untuk 256-*gray-level*), dengan 0 adalah hitam dan 255 adalah putih. Karena hanya terbatas 1 byte saja maka untuk mempresentasikan nilai pixel cukup 8 bit saja. *Gray-scale* adalah gambar yang memiliki *gray-level* sebagai nilai dari tiap pixel-nya.

Grayscaling adalah proses perubahan nilai pixel dari warna (RGB) menjadi *gray-level* (Prasetyo, 2011). Pada dasarnya proses ini dilakukan degan meratakan nilai pixel dari 3 nilai RGB menjadi 1 nilai. Untuk memperoleh hasil yang lebih baik, nilai pixel tidak langsung dibagi menjadi 3 melainkan terdapat persentasi dari masing-masing nilai. Salah satu persentasi yang sering digunakan adalah 29,9% dari warna merah (Red), 58,7% dari warna hijau (Green), dan 11,4% dari warna biru (Blue). Nilai pixel didapat dari jumlah persentasi 3 nilai tersebut.

Untuk proses *grayscaling* dapat digunakan beberapa rumus sebagai berikut[14].

1. **Lightness**

   Graylevel akan di hitung sebagai berikut

   $$\text{Lightness} = {}^{1}/_{2} \times (\max(R,G,B) + \min(R,G,B)) \quad \text{.....(2.1)}$$

2. **Luminosity**

   Graylevel akan di hitung sebagai berikut

   $$\text{Luminosity} = (R \times 0.21)+(G \times 0.75)+(B \times 0.07) \quad \text{..... (2.2)}$$

3. **Average**

   Graylevel akan di hitung sebagai berikut

   $$\text{Average} = (R + G + B) \div 3 \quad \text{.....(2.3)}$$

Hasil dari ke tiga persamaan tersebut dapat dilihat pada gambar 2.14.



Gambar 2. 14 Hasil grascaling dari masing-masing rumus

Dalam penelitian ini penulis mengubah citra RGB menjadi citra keabuan (*grayscale*) menggunakan rumus pada persamaan (2.2), yaitu .*luminocity*.

2.6 Normalisasi

Normalisasi merupakan salah satu proses dalam tahap *preprocessing* citra yang dilakukan sebelum masuk ke proses pendeteksian. Proses untuk merubah atau menormalkan tingkat intensitas cahaya pada kedua citra tersebut, citra dipetakan pada *pixel* dengan ukuran tertentu sehingga memberikan representasi dimensi yang tetap. Bertujuan untuk menyesuaikan data citra pada basis data . proses normalisasi disesuaikan dengan kebutuhan pada proses

pendeteksian yang digunakan[12]. Proses yang digunakan pada tahap normalisasi ini mengggunakan konsep normalisasi sesuai dengan metode *Normalized Coss Corelation,* dimana dalam tahapan ini meliputi, menghitung *mean* (rata-rata) untuk citra uji dan citra template, dan menghitung nilai *standard deviation* (menormalkan intensitas cahaya pada kedua citra).

Rumus untuk menghitung *mean* sebagai berikut :

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{\mathrm{n}} \quad \text{.....(2.2)}$$

Sedangkan rumus untuk menghitung *standard deviation* sebagai berikut :

$$\hat{f} = \frac{\mathrm{f} - \bar{f}}{\sqrt{\sum (f - \bar{f})^2}} \qquad \hat{g} = \frac{\mathrm{g} - \bar{g}}{\sqrt{\sum (g - \bar{g})^2}} \quad \text{.....(2.3)}$$

Keterangan :

f = Template

g = image target

$\bar{f}$ = *mean* dari f

$\bar{g}$ = *mean* dari g

$\hat{f}$ = normalisasi dari f

$\hat{g}$ = normalisasi dari g

2.7 Convolution

Konvolusi (*convolution*) adalah sebuah proses dimana citra dimanipulasi dengan menggunakan ekstnal mask / subwindows untuk menghasilkan citra yang baru. Dimana ekstnal mask konvolusi dinyatakkan dalam bentuk matriks yang ukuranya lebih kecil dari ukuran citranya. Setiap elemen matrik disebut koefisien konvolusi. Konvolusi pada fungsi dua dimensi dapat didefinisikan pada persamaan 2.4.

$g(x, y)\, f(x, y) * h(x, y) \sum_{a=-a}^{a} \sum_{b=-a}^{a} f\ (a, b) h(x\ a, y\ b)$ .....(2.4)

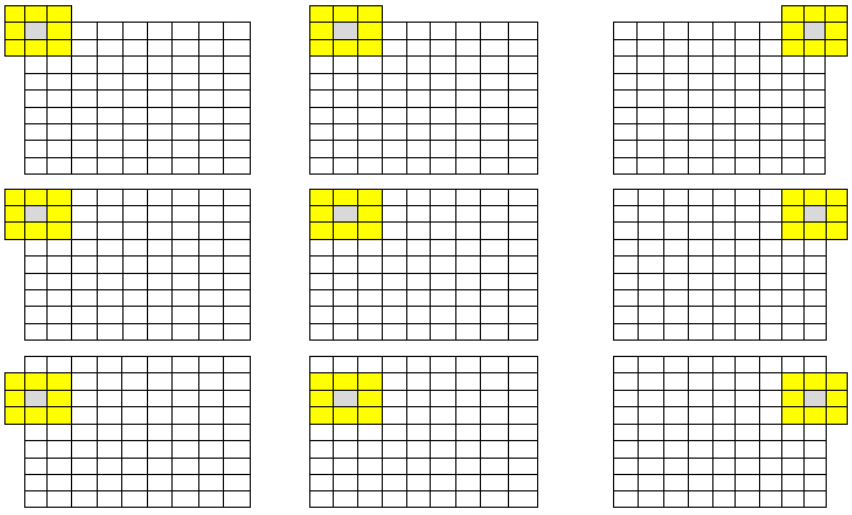
Ekstnal mask pada operasi konvolusi secara umum menggunakan metode *prewwit*, metode *sobel*, metode *Robert* dan metode-metode lainya. Dalam penelitian ini proses konvolusi yang dilakukan menggunakan metode *Normalized Cross Corelation, dimana* ekstnal mask yang digunakan menggunakan *template* dari hasil normalisasi.

Operasi konvolusi dilakukan dengan menggeser penapis(window) *pixel* per *pixel* dan hasilnya disimpan dalam matrik yang baru. Berikut ini ilustrasi operasi konvolusi :

$$\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \quad * \quad \begin{bmatrix} p1 & p2 & p3 \\ p3 & p5 & p6 \\ p7 & p8 & p9 \end{bmatrix} \quad f(x,y)$$

Template(penapis) Citra Uji(target)

Ilustrasi proses perkalian antar dua buah matrik tersebut dapat dilihat pada gambar 2.15.



Gambar 2. 15 Ilustrasi teknik konvolusi matrik

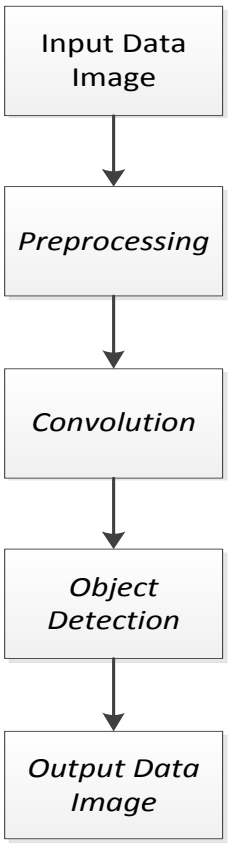
2.8 Deteksi Objek

Deteksi objek dalam *Digital Image Processing*, suatu proses yang digunakan untuk menentukan keberadaan objek tertentu di dalam suatu citra

digital. Proses deteksi tersebut dapat dilakukan dengan berbagai macam metode yang umumnya melakukan pembacaan fitur-fitur dari seluruh citra *input* tersebut yang akan dibandingkan dengan fitur dari objek referensi atau *template* yang dimaksud atau tidak.

Aplikasi deteksi objek adalah aplikasi yang dapat mencari keberadaan objek yang dicari pada citra *input* sesuai dengan citra *template* atau citra yang menjadi acuan.

Hasil dari proses deteksi objek adalah berupa citra yang sudah diberikan border pada citra *input*, sesuai dengan citra atau objek yang dicari, dimana tingkat keakuratan pendeteksian objek tersebut diukur dengan ditemukan atau tidaknya suatu objek yang dicari pada aplikasi yang dibuat. Secara umum blok diagram kerja deteksi objek dapat dilihat sebagai berikut[14].



Gambar 2. 16 Blok Diagram Kerja Sistem

2.8.1 Input Data Image

Input data image merupakan proses mengunggah data citra uji dan citra *template*.

2.8.2 Preprocessing

Preprocessing merupakan suatu proses untuk menghilangkan bagian-bagian yang tidak diperlukan pada gambar input untuk proses selanjutnya, proses ini meliputi proses :

1. Konversi(memecah) dari RGB kedalam masing-masing warna R(Red),G(Green), dan B(Blue).
2. *Grayscale* merupakan proses mengubah citra warna menjadi citra keabuan, hal ini digunakan untuk menyederhanakan model citra.
3. Normalisasi adalah proses untuk merubah atau menormalkan tingkat intensitas cahaya pada kedua citra tersebut, bertujuan untuk meningkatkan akurasi pendeteksian pada citra yang akan di uji. Proses yang digunakan pada tahap normalisasi ini mengggunakan konsep normalisasi sesuai dengan kebutuhan pada metode *Normalized Coss Corelation.*

2.8.3 Convolution

Konvolusi pada umumnya adalah sebuah proses dimana citra dimanipulasi dengan menggunakan ekstnal mask / subwindows untuk menghasilkan citra yang baru. Sedangkan filtering tanpa menggunakan mask tetapi hanya menggunakan *pixel* tetangga untuk mendapatkan *pixel* yang baru. Pada tahapan konvolusi ini digunakan hanya untuk mencari nilai korelasi dari perkalian dua buah matrik.

2.8.4 Object Detection

Deteksi objek merupakan proses yang dilakukan untuk mencari nilai kecocokan antar citra target dan citra yang dijadikan *template*. Dimana hasil dari

deteksi objek akan ditunjukan titik koordinat yang sesuai dengan letak dari objek yang dicari atau *template* pada citra target.

2.8.5 Output Data Image

Output yang dihasilkan dari pedeteksian objek adalah berupa citra yang sudah terdeteksi sesuai *template* dengan memberikan border berwana merah pada titik atau letak dari hasil deteksi dan menampilkan informasi seperti *Higest Value* (HV) dan *Higest Value Location* (HVL). Dimana HV adalah hasil dari perhitungan korelasi antara *template* dan citra target berupa nilai terbesar, yang artinya hasil nilai tersebut adalah nilai yang *macth* untuk *template* yang dicari. Sedangkan *Highest Value Location* (HVL) yaitu lokasi *pixel(i,j)* dimana objek yang dicari berada pada titik *pixel* tersebut pada citra target.

2.9 Threshold

Thresholding merupakan salah satu teknik dalam *image recognition* dalam sebuah pendeteksian objek yang berbasis *template.* Dalam perbedaan nilai intensitas yang signifikan antara citra uji (target) dan citra utama atau *template* (referensi) yang digunakan. Dalam pelaksanaan *thresholding* membutuhkan suatu nilai yang digunakan sebagai nilai pembatas antara objek utama dengan citra target, dan nilai tersebut dinamakan *threshold.*

Threshold digunakan untuk mempartisi citra dengan mengatur nilai intensitas semua piksel yang lebih dari *threshold T* sebagai latar depan dan yang lebih kecil dari nilai *threshold T* dilakukan berdasarkan histogram *grasyscale*[3].

Teknik dalam proses *threshold* yang digunakan.

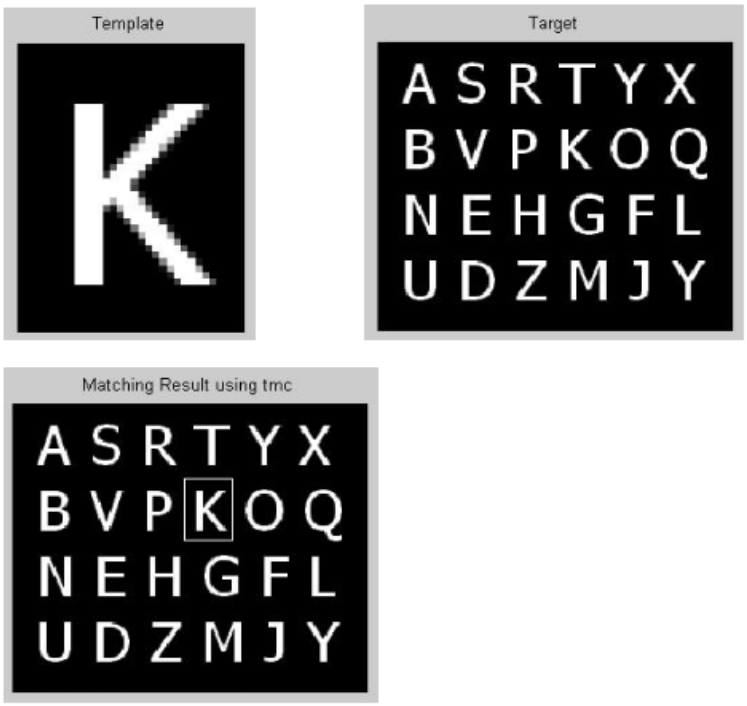
1. max ← mencari nilai maksimal dari hasil *Cross-Correlation* pada setiap piksel.
2. thresh ←menentukan nilai *threshold.*
3. hasil ← menghitung proses perkalian nilai max dan thresh.

4. Jika hasil lebih kecil atau kurang dari nilai *threshold* maka objek yang dicari tidak terdeteksi.

2.10 Template Matching

Template matching merupakan salah satu teknik dalam pengolahan citra yang berfungsi untuk mencocokan tiap-tiap bagian dari suatu citra degan citra yang menjadi *template* (acuan)[10]. Pada dasarnya proses yang dilakukan dengan cara memasukan citra yang mengandung *template* tertentu yang kemudian dibandingkan dengan *template* pada basis data. Template ditempatkan pada pusat bagian citra yang akan dibandingkan dan dihitung seberapa banyak titik yang paling sesuai dengan *template.* Langkah ini di ulangi terhadap keseluruhan citra masukan yang akan dibandingkan

Nilai kesesuaian titik yang paling besar antara citra masukan dan citra *template* menandakan bahwa *template* tersebut merupakan citra *template* yang paling sesuai masukan. Algoritma atau metode *template matching* sering digunakan untuk mengidentifikasi citra karakter huruf, angka, sidik jari, (*fingerprint*) dan aplikasi-aplikasi pencocokan citra lainya.



Gambar 2. 17 Ilustrasi *Template Matching*

Gambar 2.17 bagian kanan merupakan citra yang mengandung berbagai macam objek yang sama dengan objek pada *template* yang ada di sebelah kiri, dan yang paling bawah adalah hasil dari sebuah pendeteksian tersebut. *Template matching* diposisikan pada citra yang akan dibandingkan dan dihitung derajat kesesuaian pola pada citra masukan dengan pola pada citra *template.*

Template Matching dapat dibagi antara dua pendekatan, yaitu :

1. Pendekatan berbasis fitur

   Pendekatan berbasis fitur menggunakan pencarian dan *template* gambar seperti tepi atau sudut, sebagai pembanding pengukuran matrik untuk menemukan lokasi *Temlate Matching* yang terbaik pada sumber gambar. Sebuah pendekatan berbasis fitur dapat dianggap sebagai pejndekatan yang lebih efektif, jika *template* gambar memiliki fitur yang kuat jika pencocokan di pencarian gambar bisa diubah dengan cara tertentu. Karena pendekatan ini tidak mempertimbangkan keseluruhan *template* gambar, komputasi dapat

lebih efisien ketika bekerja dengan sumber gambar beresolusi lebih besar, sebagai pendekatan alternatif, berbasis *template*, mungkin memerlukan pencarian titik-titik yang berpotensi untuk menentukan lokasi pencocokan yang terbaik.

2. Pendekatan berbasis *template*

   Untuk template tanpa fitur yang kuat, atau ketika sebagaian besar *template* gambar merupakan gambar yang cocok, sebuah pendekatan berbasis *template* mungkin efektif. Seperti disebutkan diatas, karena berbasis *template*, *Template Matching* berpotensi memerlukan *sampling* dengan mengurangi resolusi pencarian dan *template* gambar oleh faktor yang sama dan melakukan operasi pada perampingan gambar yang dihasilkan (multiresolusi, atau piramida, pengolahan citra), menyediakan pencarian titik data, atau kombinasi keduanya.

2.11 Metode Normalized Cross Corelation

Pada penelitian ini *Template Matching* akan menggunakan algoritma *Normalized Cross Corelation* (NCC) yang dikembangkan oleh JP. Lewis (Lewis, 1995). Korelasi adalah alat penting dalam pengolahan citra, pengenalan pola, pendeteksian objek dan bidang lainya. Korelasi antara dua sinyal (*cross corelation*) adalah pendekatan standar untuk mendeteksi fitur dalam suatu citra.

Teknik *Template Matching* mencoba untuk menjawab beberapa variasi pertanyaan berikut : Apakah gambar vberisi *view* tertentu dari beberapa fitur ?, jika demikian, dimana?. Penggunaan *Cross-Corelation* untuk pencocokan *template* termotivasi oleh ukuran jarak *(Euclidean distance).*

$$d^2_{f,t}(\mathrm{u,v}) = \sum_{x,y}[f(x,y) - t(x-u, y-v)]^2 \quad \text{.....(2.5)}$$

(penjumlahan atas x,y dengan dengan posisi fitur ada pada u,v) atau jika d^2 diperluas menjadi

$$d^2_{f,t}(\mathrm{u,v}) = \sum_{x,y}[f(x,y) - 2f(x,y)t(x-u, y-v) + t^2(x-u, y-v)]$$

$\sum t^2(x-u, y-u)$selalu bernilai konstan. Jika $\sum t^2(x, y)$mendekati nilai kontan, maka tersisa *Cross-Correlation*

$$c(u, v) \quad = \sum_{x,y}[f(x, y)t(x-u, y-v)] \text{.....(2.6)}$$

Yang merupakan ukuran tingkat kemiripan antara fitur dengan citra.

Untuk mendapatkan titik-titik yang sama antar citra, maka dapat digunakan *correlation-based*. Dalam *correlation-based* kita memotong-motong citra yang pertama dengan ukuran tertentu pada bagian sudut kemudian per potongan kita kemudian membandingkan *pixel* per *pixel* dengan citra yang kedua, dimana citra yang kedua tetap utuh, untuk mencari letak persamaan dari kedua citra tersebut.

Jika energi citra yang dinyatakan dengan $\sum f^2$(x,y) ternyata tidak bernilai konstan, pencocokan fitur dengan *cross-correlation* akan gagal. Sebagai contoh, korelasi antara *template* dan area pencocokan yang sama persis dalam citra mungkin lebihsedikit dibandingkan dengan korelasi antara *template* dengan *spot* yang terang. Kelemahan lain dari teknik *template matching* adalah bahwa jangkauan *c(u,v)* tergantung pada ukuran *template* dan tingkat pencahayaan pada citranya.

Variasi dalam energi citra *template* dapat dikurangi dengan filter *high-pass* sebelum *Cross-Correlation.* Pada implementasi sebuah transformasi domain proses filter dapat dengan mudah ditambahkan ke proses frekuensi domain, meskipun pemulihan dalam memotong frekuensi jadi masalah tersendiri, pemotongan frekuensi yang terlalu rendah dapat membuat signifikan variasi energi citra, sebaliknya jika pemotongan frekuensi terlalu tinggi dapat menghapus informasi yang berguna pada saat proses pencocokan.

Normalized Cross Corelation adalah salah satu solusi yang digunakan untuk *template matching,* dapat mengatasi masalah ini dengan menormalisai vektor citra dan *template* menjadi satuan panjang, dan menghasilkan koefisien korelasi yang tepat[8].

Cross correlation merupakan perkalian antara dua buah matrik yang berbeda yang digunakan untuk mengkategorikan data berdasarkan nilai yang sama dengan rumus sebagai berikut :

$$\text{NCC (f,g)} = cfg\left(\hat{f}.\hat{g}\right) = \sum_{[i,j]\in R} \hat{f}(i.j)\ \hat{g}(i.j) \quad \text{.....(2.7)}$$

f = Template

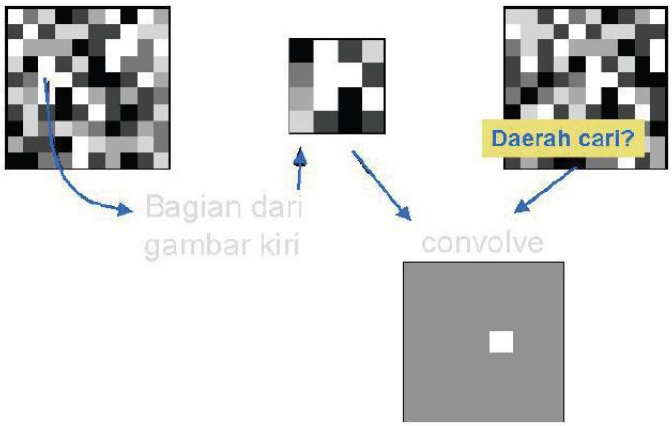
g = image target

cfg = korelasi antara f dan g

$\hat{f}$ = normalisasi dari f

$\hat{g}$ = normalisasi dari g

Karena metode korelasi yang digunakan adalah *Normalized Cross Corelation* maka perlu adanya normalisasi terlebih dahulu pada window citra yang akan dicari korelasinya dengan persamaan 2.3.



Gambar 2. 18 Proses Korelasi

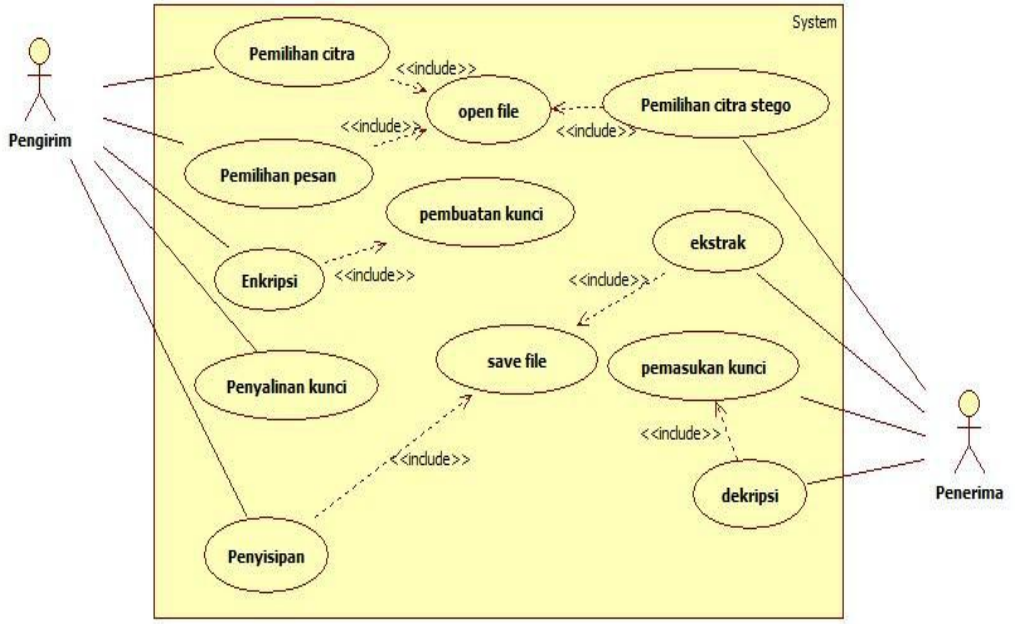
Dimana $\bar{f}$ merupakan nilai *mean* dari *template* dan $\bar{g}_{(i,j)}$ merupakan nilai *mean f(i,j)* dalam area suatu *template.*

2.12 *Unified Modeling Language* (UML)

Unified Modeling Language(UML) merupakan sistem arsitektur yang bekerja dalam OOAD(*object-Oriented Analysis/Design*) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkontruks, dan mendokumentasikan *artifact* (suatu informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software, dapat berupa model, deskripsi, atau software) yang terdapat dalam sistem[6].

2.12.1 *Use CaseDiagram*

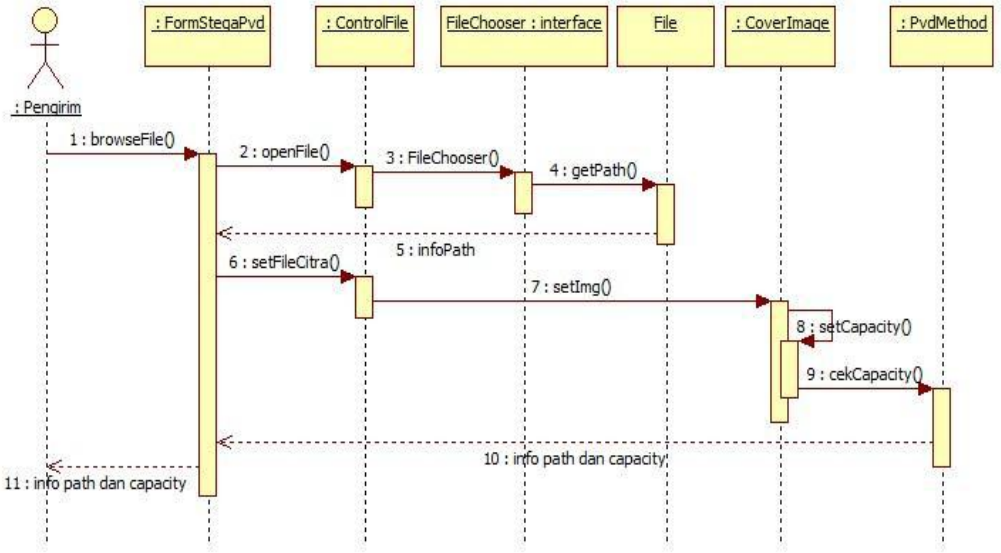
Use case diagram menggambarkan tujuan penting fungsionalitas sistem yang akan ditampilkan pada user. Sebua use case merepresentasikan sebuah interaksi antara actor dengan sistem. Use case diagram dapat membantu dalam penyusunan *requirement* sebuah *sistem,* mengkomunikasikan rancangan dengan klien, dan merancang test case untuk semua fitur yang ada pada sistem.



Gambar 2. 19 Contoh *Use Case* Diagram

2.12.2 *SequenceDiagram*

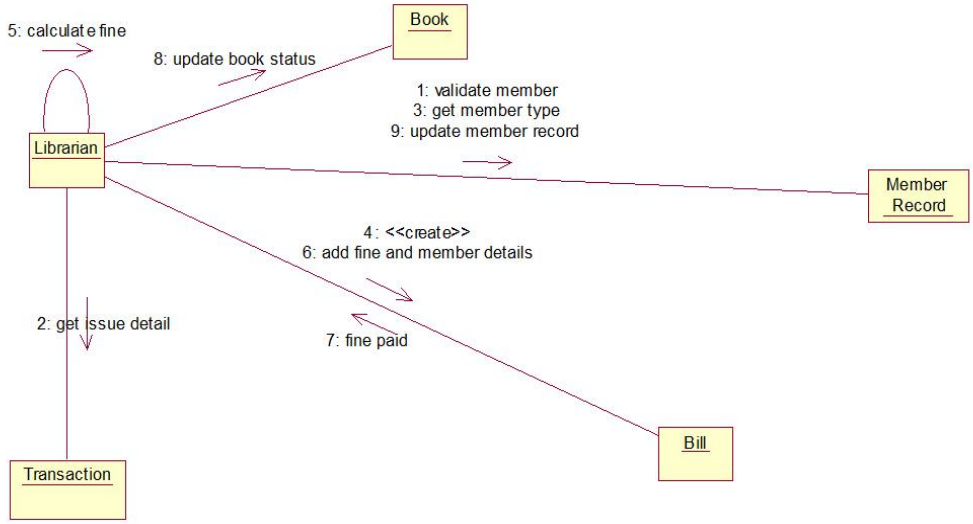
Sequence diagram biasanya digunakan untuk menggambarkan skenario atau rangkaian lengkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan *output* tertentu. Diawali dari apa yang mend-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk *actor*, memiliki *lifeline vertical*.*Message* digambarkan sebagai garis berpanah dan satu objek ke objek lainnya. Pada fase desain berikutnya *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message.*



Gambar 2. 20 Contoh *Squance Diagram*

2.12.3 Collaboration Diagram

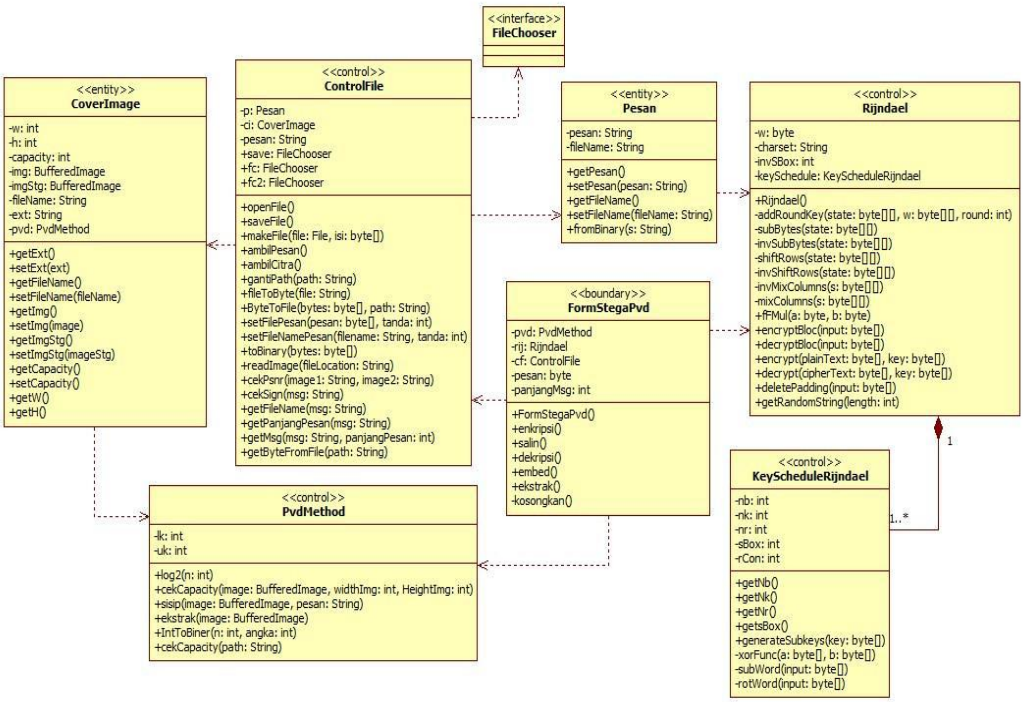
Collaboration diagram dipakai untuk memodelkan interaksi antar objek di dalam sistem. Berbeda dengan *sequence diagram* yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan, *Collaboration* diagram lebih fokus pada pemahaman atas keseluruhan operasi yang dilakukan oleh objek.



Gambar 2. 21 Contoh *Collaboration Diagram*

2.12.4 Class Diagram

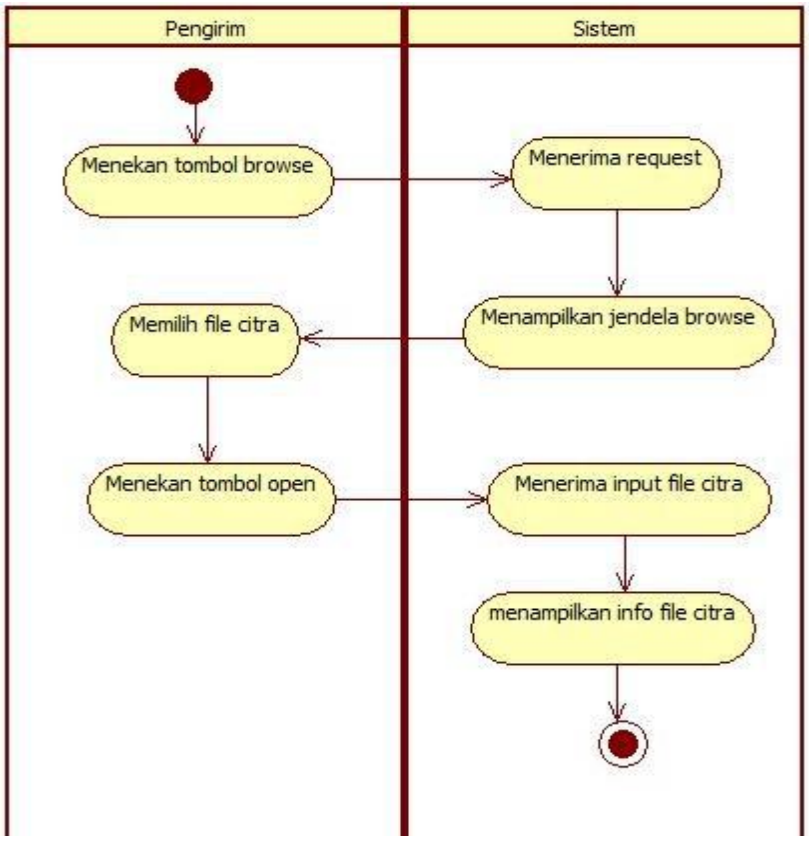
Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class diagram* menunjukan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan.



Gambar 2. 22 Contoh *Class Diagram*

2.12.5 *ActivityDiagram*

Activity diagram digunakan untuk menggambarkan aliran aktivitas baik dalam proses bisnis maupun *use case*. Diagram ini juga dapat digunakan untuk memodelkan aksi yang akan dilakukan saat sebuah operasi dieksekusi dan memodelkan hasil dari aksi tersebut.



Gambar 2. 23 Contoh *Activity Diagram*

2.13 JAVA

Java merupakan bahasa pemrograman berorientasi objek dan bebas *platform*[5], dikembangkan oleh *SUN Micro* System dengan sejumlah keunggulan yang memungkinkan Java dijadikan sebagai bahasa pengembangan *enterprise*. Berikut ini beberapa keunggulan dari Java yaitu :

1. Berorientasi Objek

Konsep pemrograman berorientasi objek tak lain dirancang agar dapat memandang pemrograman sebagai suatu kehidupan nyata. Ini membuat pengembangan software menjadi lebih mudah, karena seolah-olah berhubungan dengan kehidupan nyata, bukan lagi kehidupan ‘dunia lain’. Java merupakan salah

satu bahasa yang memiliki dukungan penuh terhadap konsep pemrograman berorientasi objek ini.

2. *Multiplatform*

Java merupakan bahasa pemrograman yang bisa dijalankan pada komputer platform yang berbeda, asalkan JVM telah terintall untuk platform itu. Jika membuat program Java di atas Windows, maka ia bisa dijalankan pada Linux, Unix, Macintosh. Selain itu, Java juga bisa digunakan untuk membuat program yang berjalan di atas berbagai mobile device (dengan J2ME), PDA, embeded system, dan lain-lain.

3. *Multithread*

Fitur multithread digunakan untuk menjalankan sejumlah proses secara bersamaan. Dengan menggunakan fasilitas ini, dapat membuat program Java yang menjalankan beberapa perintah sekaligus, sehingga tidak perlu menunggu sebuah perintah selesai dikerjakan untuk mengerjakan perintah lain.

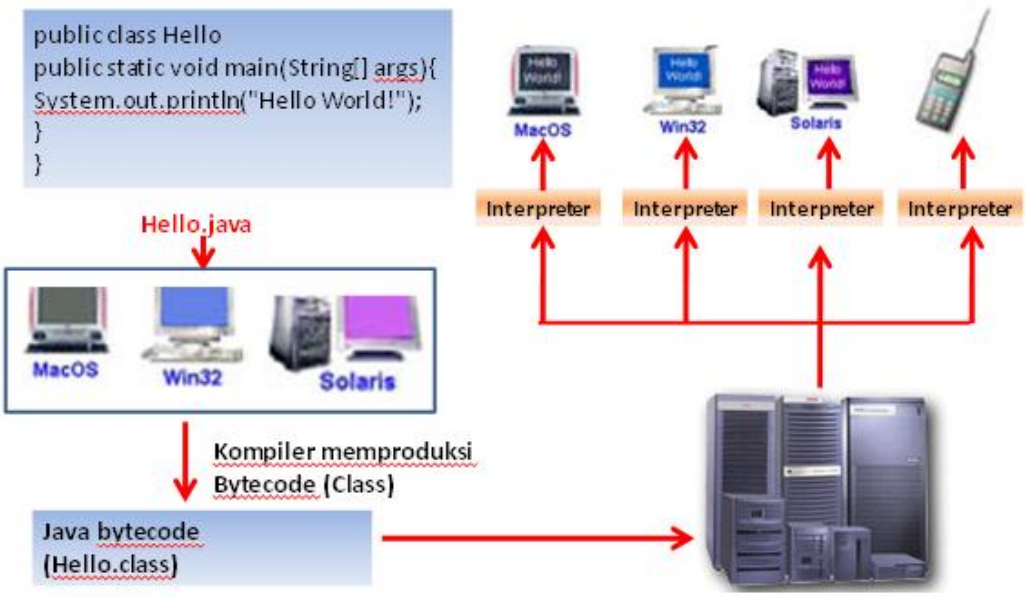
2.13.1 JVM dan *Byte Code*

Mengapa sebuah program Java bisa dijalankan pada semua sistem operasi? Jawabannya adalah karena keberadaan *Java Virtual Machine* (JVM) dan konsep byte-code. Karena hal inilah mengapa ada slogan khusus untuk Java, *write one, run anywhere*. Artinya hanya dengan membuat dan mengkompile sebuah program Java, program tersebut bisa dijalankan dimana saja.

Program Java yang ingin dijalankan (termasuk program Java lain yang diperlukan oleh program tersebut) harus dikompile terlebih dahulu dengan javac. Proses kompilasi program Java akan menghasilkan sejumlah file class dalam bentuk *byte-code*, file dengan ekstension ".class".

Byte code Java tidak bisa dijalankan seperti *binary code*. *Byte code* harus dijalankan pada JVM, yaitu mesin Java yang mengerti perintah-perintah yang ada *byte code*. JVM inilah yang selanjutnya berhubungan dengan sistem operasi untuk menjalankan perintah pada *byte code* tersebut Untuk setiap sistem operasi

memiliki JVM yang berbeda. JVM harus sudah terinstall pada sistem agar dapat menjalankan program Java yang berbentuk *byte code* itu. Gambar 2.25, memperihatkan bagaimana sebuah file Java bisa dijalankan pada sistem operasi yang berbeda-beda, misalnya Windows, Solaris dan Macintos.



Gambar 2. 24 Java dengan Sistem Operasi

1. Source code Hello.java dapat dijalankan oleh berbagai macam sistem operasi. contoh: MacOS
2. Kompiler akan memproduksi Bytecode (Class) dari file Hello.class
3. Java bytecode (Hello.class) akan di kompilasi dan dijalankan langsung oleh interpreter
4. Akhirnya, Program Hello.java dapat dijalankan diberbagai platform.

2.13.2 J2SE (Standar Edition)

Yaitu fitur teknologi java yang dirancang untuk berjalan diatas PC dan workstation yang dapat berjalan diplatform sistem operasi linux, macintosh, windows, dll. SUN microsystem secara resmi membagi J2SE menjadi dua bagian besar yaitu J2SE *core* dan J2SE desktop. J2SE *core* memiliki teknologi turunan yang meliputi teknologi security, teknologi debugging, teknologi dokumentasi,

teknologi database ,dan banyak teknologi turunan lainnya. Sedangkan J2SE desktop memiliki beberapa teknologi meliputi JRE (*Java Runtime Environment*), JFC (*Java Foundation Classes*), *Java Sound* API, dll. Berikut ini adalah langkah-langkah dalam menghasilkan suatu program dengan bahasa java, yaitu :

1. Mengkompile Program Java

Kompiler yang digunakan untuk mengkompile semua program Java, baik program aplikasi atau applet, adalah javac. Program ini terdapat pada direktori bin dari direktori dimana J2SDK diinstall. Sintaks untuk mengkompile program Java dengan javac:

javac <nama-file-java.java>

2. Menjalankan Program Java

Java memiliki 2 program untuk menjalankan program Java, yaitu: java dan appletviewer. Java digunakan untuk menjalankan program aplikasi dan appletviewer digunakan untuk menjalankan applet. Program java dan appletviewer akan menjalankan Java Virtual Machine (JVM) untuk menjalankan kedua bentuk program Java.

a. Java

Sintaks untuk menjalankan program Java (*byte code*) dengan java:

java <nama-file-java>

Perhatikanlah bahwa pada saat menjalankan program Java tidak boleh mengikutkan nama ekstensi dari file class (.class).

b. Appletviewer <nama-file-html>

Sintaks untuk menjalankan applet dengan appletviewer:

appletviewer <nama-file-html>

2.14 Netbeans

Neetbeans IDE adalah IDE open source yang ditulis sepenuhnya dengan bahasa pemograman Java menggunakan platform NetBean. Netbeans IDE mendukung pengembangan semua tipe apliaksi Java (J2SE, web, EJB, dan aplikasi mobile). Fitur lainya adalah sistem proyek berbasis Ant, kontrol versi, dan refactoring.

Pada versi NetBeans IDE 8.0.1 yang dirilis Mei 2008 telah disediakan fitur-fitur Jaava EE yang termasuk sudah ada (termasuk Java Persistence support, EJB-3 dan JAX-WS). Sementara paket tambahanya, NetBeans Enterprise Pack mendukung pengembangan aplikasi perusahaan Java EE 5, meliputi alat desain visual SOA, skema XML, web service dan pemodelan UML. NetBeans C/C++ Pack mendukung proyek C/C++.

Semua fungsi IDE disediakan oleh modul-modul. Tiap modul menyediakan fungsi yang didefinisikan dengan baik, seperti dukungan untuk bahasa pemograman Java, editing, atau dukungan bagi CVS.