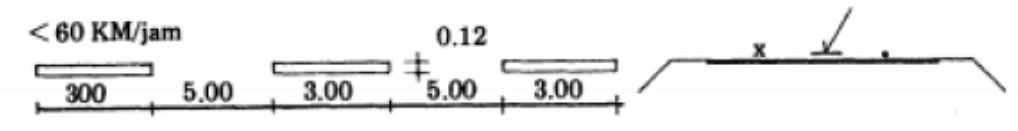
# BAB 2
# LANDASAN TEORI

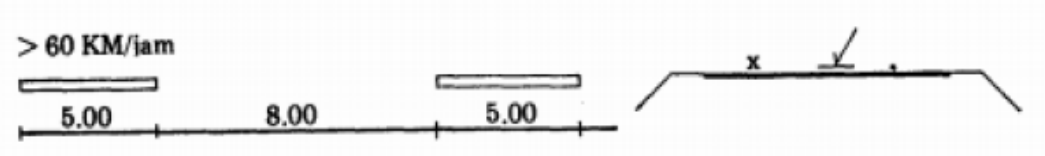
## 2.1 Pengertian Marka Jalan

Marka jalan merupakan suatu penanda bagi para pengguna jalan untuk membantu kelancaran jalan dan menghindari adanya kecelakaan. Pada umumnya marka jalan berfungsi untuk memisahkan antara lajur satu dengan yang lainnya, atau memisahkan lajur yang berlawanan.

Marka jalan memiliki jenis-jenis garis yang berbeda dengan fungsi yang berbeda-beda pula. Sebagai contoh adalah garis terputus yang berfungsi sebagai garis sumbu dan pemisah pada jalan. Berikut adalah jalan dua lajur dan dua lajur dapat dilihat pada Gambar 2.1 dan Gambar 2.2:

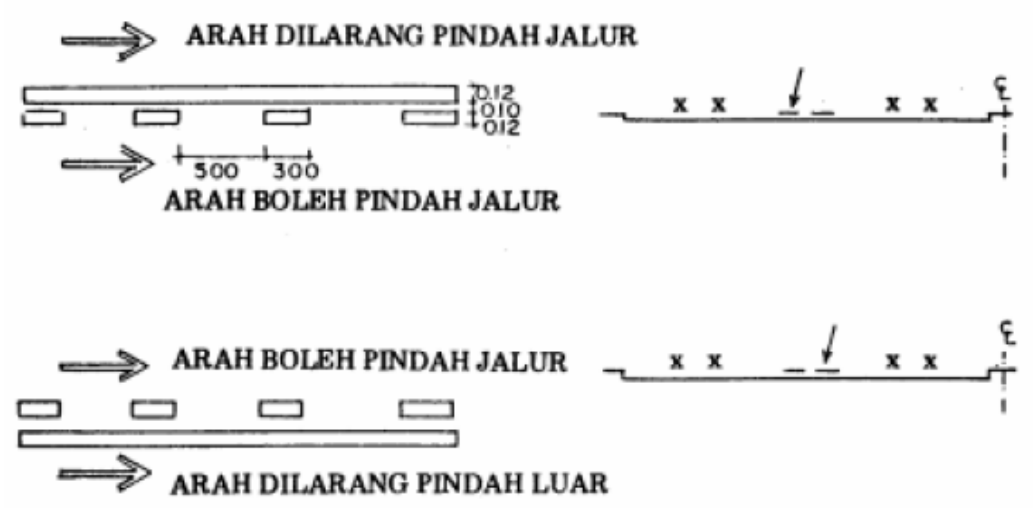



**Gambar 2.1** Jalan 2 Jalur dan 2 Lajur dengan Lebar > 5,50 m




**Gambar 2.2** Jalan Lebih dari 2 Lajur

Contoh lainnya yaitu garis penuh yang berfungsi sebagai garis dilarang pindah jalur dari salah satu sisi seperti Gambar 2. SEQ Gambar_2. * ARABIC 3 [**5]**:




**Gambar 2. SEQ Gambar_2. * ARABIC 3** Arahan Marka Dilarang dan Boleh Pindah Jalur

## 2.2 Karakteristik Video Digital

*Video digital* pada dasarnya tersusun atas serangkaian *frame*. Rangkaian *frame* tersebut ditampilkan pada layar dengan kecepatan tertentu, bergantung pada laju *frame* yang diberikan ( dalam *frame* / detik ). Jika laju *frame* cukup tinggi, mata manusia tidak dapat menangkap gambar per *frame*, melainkan menangkapnya sebagai rangkaian yang kontinyu. Masing - masing *frame* merupakan gambar / citra ( *image* ) *digital*. Suatu *image digital* direpresentasikan dengan sebuah matriks yang masing - masing elemennya merepresentasikan nilai intensitas. Jika I adalah matriks dua dimensi, I (x,y) adalah nilai intensitas yang sesuai pada posisi baris x dan kolom y pada matriks tersebut. Titik - titik dimana *image* di*sampling* disebut sebagai *picture elements* atau sering dikenal sebagai *pixel*. Karakteristik *video digital* ditentukan oleh resolusi ( *resolution* ) atau dimensi *frame ( frame dimension ),* kedalaman *pixel ( pixel depth ),* dan laju *frame ( frame rate ).* Karakteristik – karakteristik ini yang akan menentukan kualitas *video* dan jumlah *bit* yang dibutuhkan untuk menyimpan atau mentransmisikannya. [ HYPERLINK \l "Aal96" **6** ]

### 2.2.1 Resolusi

Resolusi ( *resolution* ) atau dimensi *frame* ( *frame dimension* ) adalah ukuran sebuah *frame* pada *video digital*. Resolusi dinyatakan dalam *pixel* x *pixel*. Semakin tinggi resolusi, semakin baik kualitas *video* tersebut, dalam arti bahwa dalam ukuran fisik yang sama, *video* dengan resolusi tinggi akan lebih detil. Namun, resolusi yang tinggi akan mengakibatkan jumlah *bit* yang diperlukan untuk menyimpan atau mentransmisikannya meningkat. Misalnya pada format QCIF ( *Quarter Common Intermediate Formats* ), resolusinya adalah 144 *pixel* x 176 *pixel*, ukuran ini umum digunakan untuk konferensi *video*.**6**]

### 2.2.2 Kedalaman Bit

Kedalaman *bit* ( *bit depth* ) menentukan jumlah *bit* yang digunakan untuk merepresentasikan tiap *pixel* pada sebuah *frame*. Kedalaman *bit* dinyatakan dalam *bit* per *pixel*. Semakin banyak jumlah *bit* yang digunakan untuk merepresentasikan sebuah *pixel*, berarti semakin tinggi kedalaman *pixel*nya, maka semakin tinggi pula kualitasnya, mengakibatkan jumlah *bit* yang diperlukan menjadi lebih tinggi. Satu byte (8 *bit*) untuk tiap *pixel*, diperoleh $2^8$ atau 256 *level* intensitas. Dengan *level* intensitas sebanyak itu, umumnya mata manusia sudah dapat dipuaskan. Ke dalam *pixel* paling rendah terdapat pada *binary-value image* yang hanya menggunakan 1 *bit* untuk tiap *pixel*, sehingga hanya ada dua kemungkinan bagi tiap *pixel*, yaitu 0 (hitam) atau 1 (putih). [ HYPERLINK \l "Aal96" **6** ]

### 2.2.3 Laju *Frame*

Laju *frame* ( *Frame rate )* menunjukkan jumlah *frame* yang digambar tiap detik, dan dinyatakan dengan *frame* per detik. Sehubungan dengan laju *frame* ini, ada dua hal yang perlu diperhatikan, yaitu kehalusan gerakan ( *smooth motion* ) dan kilatan (*flash*). Kehalusan gerakan ditentukan oleh jumlah *frame* yang berbeda per detik. Untuk mendapatkan gerakan yang halus, *video digital* setidaknya harus menampilkan sedikitnya 25 *frame* per detik. Kilatan ditentukan oleh jumlah berapa kali layar digambar per detik. Dengan 20 *frame* per detik, kilatan sudah dapat dilenyapkan. *Video* yang berkualitas baik akan memiliki laju

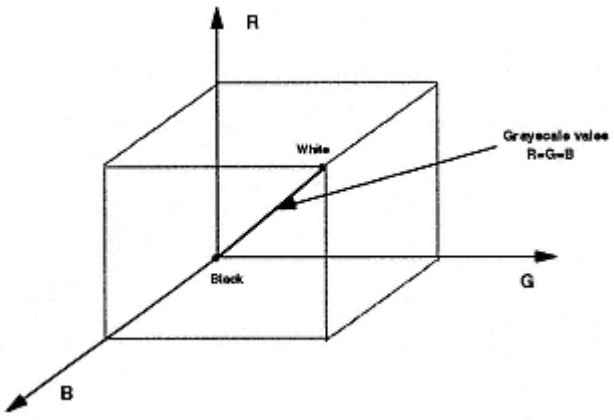
*frame* yang tinggi, minimal 30 *frame/*detik, setidaknya sesuai dengan mata manusia, yang berarti membutuhkan jumlah *bit* yang lebih tinggi.**6**]

### 2.2.4 Representasi Warna

Pada *video digital*, umumnya data *video* dipisahkan menjadi komponen – komponen, baik komponen warna maupun komponen kecerahan. Penyajian semacam ini disebut *component video*. Pada *component video*, tiap komponen dipisahkan dengan cara tertentu. Beberapa cara pemisahan komponen tersebut adalah [ HYPERLINK \l "Aal96" **6** ]:

1. RGB

   Data *video* dapat dipisahkan menjadi komponen – komponen untuk masing – masing warna, yaitu merah ( *Red* ), hijau ( *Green* ), dab biru ( *Blue* ). Warna tiap *pixel* ditentukan oleh kombinasi intensitas dari masing – masing komponen warna. Sebagai contoh, pada RGB 24 *bit*, masing – masing komponen warna dinyatakan dalam 8 *bit* atau 256 *level*. Berikut adalah representasi RGB pada warna gambar, dapat dilihat pada Gambar 2.3:




**Gambar 2.3** Representasi RGB pada Warna Gambar

2. YUV

   Pemisahan komponen tidak hanya dilakukan dengan pemisahan warna, namun dapat juga dilakukan dengan memisahkan menurut komponen kecerahan ( *luminance* ) dan komponen warna ( *crominance* ). Pada format PAL ( *Phase Alternating Line* ), sinyal kecerahan dinyatakan dengan Y, sedangkan dua sinyal warna dinyatakan dengan U dan V.

3. YIQ

Pemisahan sinyal *video* menjadi komponen kecerahan dan komponen warna dapat dilakukan juga sesuai dengan format NTSC ( *National Television System Committee* ), komponen kecerahan dinyatakan dengan Y, dan dua komponen warna dinyatakan dengan I dan Q. Karena persepsi mata manusia lebih peka pada kecerahan dari pada warna, maka *crominance* cukup di-*sampling* separuh dari *luminance.*

## 2.3 Grayscale

*Grayscale* merupakan range dengan warna yang abu-abu yang hanya memiliki satu nilai kanal pada setiap pixelnya, dengan kata lain nilai bagian *RED = GREEN = BLUE*. Nilai tersebut digunakan untuk menunjukan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan kebuan disini merupakan warna abu dengan berbagai dari hitam hingga mendekati putih. *Grayscale* memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan). Rentang nilai warna – warna pada RGB untuk Red (merah) rentangnya (255,0,0), Green (hijau) rentangnya (0,255,0), Blue (biru) rentangnya (0,0,255), dan Mediun Gray (abu) rentangnya (127,127,127) Untuk mendapatkan nilai keabuan digunakan persamaan sebagai berikut [**7**]:

$$0.299 \times R + 0.587 \times G + 0.144 \times B \qquad \text{(Pers 2.1)}$$

*Grayscaling* merupakan teknik mengubah citra warna menjadi citra abu-abu. Teknik ini merupakan proses awal dalam pengolahan citra. Dalam pemrograman MATLAB teknik *grayscaling* dapat digunakan dengan menggunakan fungsi *rgb2gray* sehingga yang awalnya citra warna dikonversi menjadi citra yang terdiri dari hitam dan putih atau biasa disebut citra *grayscale.* Teknik ini merupukan proses awal dalam pengolahan citra agar dapat melanjutkan pada proses berikutnya. Berikut adalah teknik grayscaling dapat dilihat pada Gambar 2.4 [**8**] :

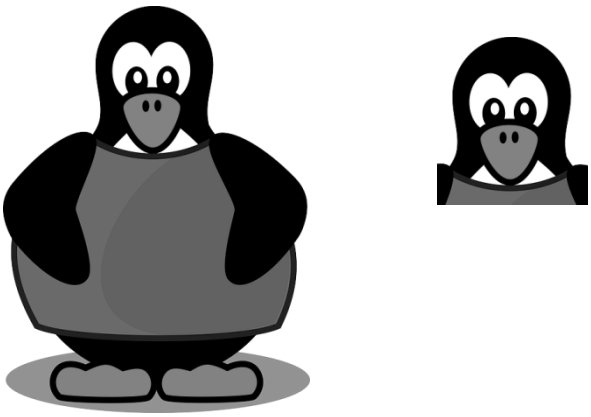


**Gambar 2.4** Teknik Grayscaling

## 2.4 Teknik *Cropping*

*Cropping* merupakan pengolahan dengan kegiatan memotong bagian dari gambar untuk mendapatkan bagian yang tertentu. [**7**]

Berikut adalah gambar proses *Cropping*, jika ingin crop hanya bagian kepala, dapat dilihat pada Gambar 2.5:



**Gambar 2.5** Gambar (a) Gambar Asli dan (b) Gambar Hasil *Cropping*

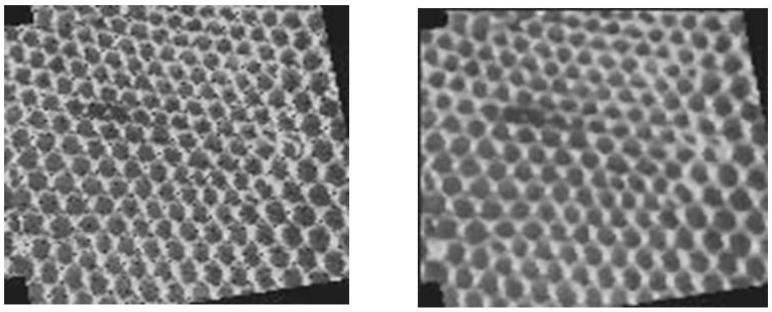
## 2.5 Median Filter

Konsep dasarnya adalah dengan menemukan nilai *pixel* yang memiliki nilai intensitas dari suatu *pixel* yang berbeda dengan nilai *pixel* yang ada didaerah sekitarnya, dan menggantinya dengan nilai yang lebih cocok. Cara yang paling sederhana dalam mencapainya adalah dengan melakukan pencegahan atau

pembatasan nilai *pixel*, sehingga suatu *pixel* tidak memiliki nilai intensitas yang diluar nilai yang ada di sekitarnya.

Untuk itu kita perlu mengetahui nilai intensitas pada suatu kelompok *pixel*. Pada pengerjaannya pada suatu daerah *pixel* seharusnya bagian yang merupakan nilai tertinggi dan terendah, dan nilai yang sebanding pada kedua bagian akhir distribusi dihilangkan. Sehingga hasilnya meninggalkan nilai *median*. Dari sana didapatkan *median filter*, dimana didapat seluruh nilai distribusi intensitas, dan dihasilkan gambar baru yang sesuai dengan nilai-nilai median yang ada.

Berikut adalah gambar awal sebelum di median filter lalu di proses dengan median filter, dapat dilihat pada Gambar 2.6:




**Gambar 2.6** (a) Gambar Asli (b) Gambar Setelah Difilter Median

Median adalah nilai tengah dari kumpulan data. Untuk mencari median dari kumpulan data yang ganjil maka:

$$X = \frac{n+1}{2} \qquad \text{(Pers 2.2)}$$

x = nilai median yang akan dicari pada suatu titik array

n= jumlah nilai pada array

Untuk *median filtering* ini, data yang digunakan untuk menghitung *median* terdiri dari kumpulan data yang ganjil. Hal ini disebabkan dengan jumlah data yang ganjil maka piksel yang akan diproses dapat berada ditengah. Pada *median filtering* digunakan matrik berdimensi N X N. Dari matrik tersebut, kemudian data yang ada diurutkan dan dimasukkan dalam sebuah matrik berukuran 1X (N X N).

Hal ini berguna untuk mempermudah menemukan *median* dari kumpulan data yang telah urut tersebut. [**9**]

## 2.6 Threshold

*Threshold* merupakan tahap untuk membuat warna citra menjadi hanya berwarna hitam atau putih. Untuk melakukan *threshold*ing terlebih dahulu harus dicari suatu nilai yang digunakan untuk batas toleransi suatu warna keabuan dapat dijadikan warna hitam atau warna putih, nilai tersebut biasa disebut nilai ambang. Setelah semua piksel diubah menjadi nilai keabuan, pencarian nilai ambangpun dilakukan. Ada beberapa cara menentukan nilai ambang yaitu pertama, dengan menentukan langsung berdasarkan nilai tengah dari level keabuan yaitu 128. Kedua, menentukan suatu nilai tertentu sebagai nilai ambang. Ketiga, menentukan nilai rata – rata dari total piksel keabuan dalam citra dan menjadikannya sebagai nilai ambang rata – rata. Kemudian setelah didapat nilai ambang tersebut maka akan dilakukan perbandingan setiap nilai piksel dari citra keabuan tersebut dengan nilai ambangnya. Jika nilai piksel lebih besar dari nilai ambangnya maka nilai piksel tersebut akan diubah menjadi 0 (piksel berwarna hitam). Sedangkan jika yang ditemukan sebaliknya yaitu nilai pikselnya lebih kecil dari nilai ambang yang telah ditentukan maka nilai piksel tersebut diubah menjadi 1 (piksel berwarna putih) [**7**]

### 2.6.1 Teknik *Threshold*ing (Pengambang-batasan)

*Thresholding* atau bisa disebut dengan pengambang-batasan yaitu sebuah teknik untuk mengubah suatu citra abu-abu menjadi citra biner (putih dan hitam). Pengembang-batasan terdapat dua jenis [**8**]:

1. Pengambang-batasan Tunggal
2. Pengambang-batasan Ganda

#### 2.6.1.1 Pengembang-batasan Tunggal

Suatu citra abu-abu diubah menjadi citra biner (putih dan hitam) dengan cara memilih suatu level keabuan T dalam citra asli, dan kemudian mengubah

setiap piksel hitam atau putih tergantung nilai keabuan lebih besar atau kurang dari T. Suatu piksel menjadi :

- Putih Jika level keabuannya >T
- Hitam jika level keabuannya <=T

Pengambang-batasan merupakan hal yang vital dalam segmentasi citra digital, dimana diinginkan untuk mengisolasi objek-objek dari latar belakang citra. Hal ini juga merupakan bidang yang penting dalam robotika. Pengambang batasasan dapat dilakukan secara sederhana dalam Matlab. Jika diasumsikan dimiliki suatu citra 8 bit dan disimpan dalam suatu variabel X, maka perintahnya X>T.

Citra yang dihasilkan kemudian dapat diproses untuk mencari jumlah bintik atau ukuran rata-rata bintik. Untuk melihat hal ini bekerja, ingat bahwa dalam Matlab, suatu operasi pada suatu angka, ketika diterapkan pada suatu matriks, diinpretasikan sebagai penerapan terhadap semua elemen-elemen matriks. Perintah X>T akan memberikan nilai balik 1 (*true*) untuk semua piksel yang memiliki intensitas keabuan yang lebih besar dari T, dan nilai balik 0 (*false*) untuk semua piksel yang memiliki intensitas keabuan yang lebih kecil atau sama dengan T.

Selain metode diatas, Matlab juga memiliki fungsi *im2bw*, yang mengambangbatas suatu citra dengan sembarang tipe data. Karena level merupakan parameter bernilai antara 0 sampai 1, yang mengindikasikan fraksi nilai-nilai keabuan yang diubah menjadi putih. Perintah ini bisa dipakai untuk citra abu-abu, citra berwarna maupun untuk citra indeks dengan tipe data uint8, uint16, ataupun double. Fungsi *im2bw* secara otomatis menskalakan nilai level menjadi suatu nilai keabuan yang cocok dengan tipe citra, dan kemudian melakukan pengambang-batasan seperti yang telah dijelaskan sebelumnya. [**8**]

#### 2.6.1.2 Pengembang-batasan Ganda

Disini dipilih dua nilai T1 dan T2 dan diterapkan suatu operasi pengambang-batasan sebagai berikut. Suatu piksel menjadi:

- Putih Jika level keabuan antara T1 dan T2

- Hitam jika level diluar rentang ini

Operasi ini bisa diimplementasikan dengan suatu variasi sederhana dari pengambang-batasan tunggal X>T1 & X<T2. Karena simbol *ampersand* berperan sebagai operasi logikal, hasilnya akan menghasilkan nilai 1 jika memenuhi kedua pertidak-samaan itu. Berikut adalah hasil *threshold*, dapat dilihat Gambar 2.7 [**8**]:



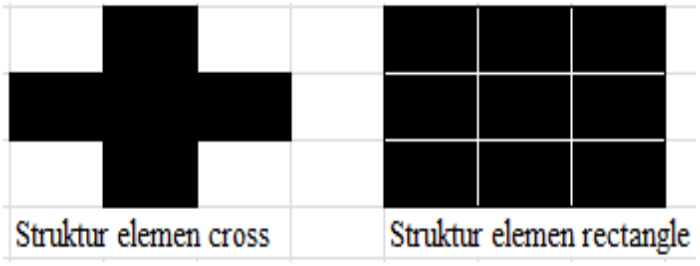
**Gambar 2.7** Gambar awal (b) Gambar setelah proses terambang-batasan

## 2.7 Operasi Morphologi

Morphologi adalah teknik pengolahan citra digital dengan menggunakan bentuk (shape) sebagai pedoman dalam pengolahan. Nilai dari setiap pixel dalam citra digital hasil diperoleh melalui proses perbandingan antara pixel yang bersesuaian pada citra digital masukan dengan pixel tetangganya. Operasi morphologi bergantung pada urutan kemunculan dari pixel, tidak memperhatikan nilai numeric dari pixel sehingga teknik morphologi sesuai apabila digunakan untuk melakukan pengolahan binary image dan *grayscale* image.

Dengan mengatur atau memilih ukuran dan bentuk dari matrik kernel (structuring element) yang digunakan maka kita dapat mengatur sensitivitas operasi morphologi terhadap bentuk tertentu (spesifik) pada citra digital masukan. Operasi morphologi standar yang dilakukan adalah proses erosi dan dilasi. Dilasi adalah proses penambahan pixel pada batas dari suatu objek pada citra digital masukan, sedangkan erosi adalah proses pemindahan/pengurangan pixel pada batas dari suatu objek. Jumlah pixel yang ditambahkan atau yang dihilangkan dari batas objek pada citra digital masukan tergantung pada ukuran dan bentuk dari structuring element yang digunakan.

Berikut adalah struktur element operasi yang dapat dipakai untuk proses operasi morphologi, dapat dilihat pada Gambar 2.8: [**9**]




**Gambar 2.8** Struktur Elemen

### 2.7.1 Dilasi

Dilasi adalah operasi morphologi yang akan menambahkan pixel pada batas antar objek dalam suatu citra digital. Atau secara rinci Dilasi merupakan suatu proses menambahkan piksel pada batasan dari objek dalam suatu image sehingga nantinya apabila dilakukan operasi ini maka image hasilnya lebih besar ukurannya dibandingkan dengan image aslinya.

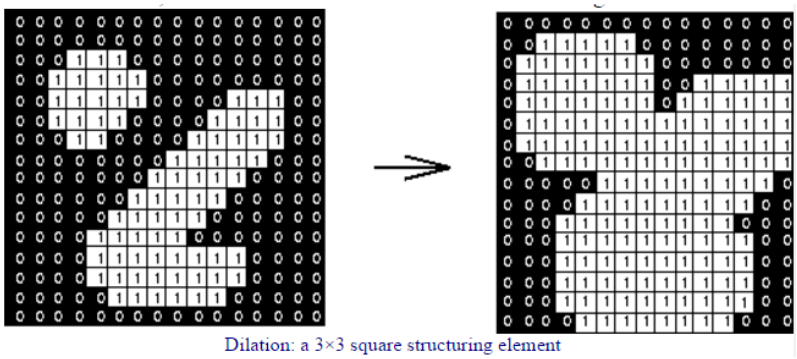
Dalam operasi dilasi, citra asli akan mengalami pelebaran dengan mengikuti bentuk *Structuring Element* yang digunakan.

Pada Gambar 2.9 dan Gambar 2.10 merupakan hasil image setelah dilakukan proses dilasi dari original image dengan stucturing element yang telah ditentukan [**10**]:




**Gambar 2.9** (a) Gambar Biner (b) Gambar Hasil Dilasi

Berikut Gambar 2.10 menjelaskan tentang proses dilasi cara membuka letak sktruktur elemen terbuka yang dipisahkan oleh wilayah tertutup.




**Gambar 2.10** Proses Dilasi

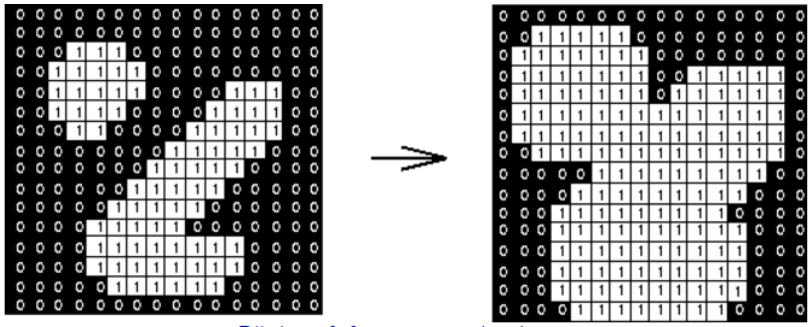
### 2.7.2 Erosi

Erosi merupakan kebalikkan dari Dilasi. Proses ini akan membuat ukuran sebuah citra menjadi lebih kecil. Berbeda dengan dilatasi, apabila erosi dilakukan maka yang dikerjakan adalah memindahkan piksel pada batasan-batasan objek yang akan di erosi. Jumlah dari piksel yang ditambah atau dihilangkan bergantung pada ukuran dan bentuk dari structuring element yang digunakan untuk memproses image tersebut. Berikut adalah proses hasil erosi, dapat dilihat pada Gambar 2.11 [**10**] :




**Gambar 2.11** (a) Gambar dari *Grayscale* (b) Gambar *Threshold* (c) Hasil Erosi

Berikut adalah Gambar 2.12 tentang proses erosi memilih bagian yang tidak perlukan pada struktur elemen:




**Gambar 2.12** Proses Erosi

### 2.7.3 Opening

*Opening* umumnya digunakan untuk menghaluskan kontur dari sebuah objek, memutuskan garis tipis yang menghubungkan dua *region* besar, dan menghilangkan tonjolan tipis. Operasi *opening* dari himpunan A oleh *Structuring Element* B, yang dinotasikan dengan $A \circ B$, didefinisikan pada persamaan berikut [**10**]:

$$(A \ominus B) \oplus B \qquad \text{(Pers 2.3)}$$

A = Matriks Awal

B = Struktur Element

$\ominus$ = Operasi “dan”

$\oplus$ = Operasi “atau”

### 2.7.4 Closing

*Closing* digunakan untuk menghaluskan bagian dari kontur tetapi berbeda dengan *opening*, *closing* biasanya menyatukan bagian kecil yang terputus dan menyatukan cekungan yang panjang dan tipis. Operasi *closing* dari himpunan A oleh *Structuring Element* B, yang dinotasikan dengan $A \cdot B$, didefinisikan pada persamaan berikut [**10**]:

$$(A \oplus B) \ominus B \qquad \textbf{(Pers 2.4)}$$

A = Matriks Awal

B = Struktur Element

$\ominus$ = Operasi “dan”

$\oplus$ = Operasi “atau”

### 2.7.5 Top-Hat dan Bottom-Hat Transformation

*Top-Hat Transform* atau *Top-Hat by Opening* didefinisikan sebagai perbedaan (*difference*) antara citra input dan hasil *opening* citra input tersebut oleh suatu *Structuring Element*.

*Bottom-Hat* atau *Top-Hat by Closing* didefinisikan sebagai perbedaan (*difference*) antara hasil *closing* citra input dan citra input itu sendiri.

Prinsip utama dari transformasi ini adalah menghilangkan objek dari sebuah citra dengan menggunakan *Structuring Element* dalam operasi *opening* dan *closing* yang tidak sesuai dengan objek yang akan dihapus. [**10**]

## 2.8 Hough transform

*Hough transform* didesain untuk menemukan garis pada suatu citra. *Hough transform* diperkenalkan oleh Paul Hough pada tahun 1962. Ide dari *Hough transform* adalah membuat persamaan dari suatu piksel dan mempertimbangkan semua pasangan yang memenuhi persamaan ini. Semua pasangan ditempatkan pada suatu larik akumulator, yang disebut larik transformasi. *Hough transform* telah dikembangkan untuk mendeteksi bentuk-bentuk umum dalam citra seperti lingkaran, elips dan parabola. Konsep dasar dari *Hough transform* adalah terdapat garis dan kurva potensial yang tak terhitung jumlahnya pada suatu citra yang melalui titik mana saja pada berbagaiu kuran dan orientasi. Transformasi dilakukan untuk menemukan garis dan kurva yang melewati banyak titik-titik di dalam citra, yaitu garis dan kurva yang terdekat dan palingsesuai dengan data pada matriks citra.

*Hough transform* menggunakan bentuk parametrik dan menggunakan pemungutan suara terbanyak (*voting*) untuk menentukan nilai parameter yang tepat. Apabila dalam citra terdapat beberapa garis yang saling berpotongan pada suatu titik, maka apabila kemudian titik tersebut ditransformasi ke dalam ruang parameter $m - c$ akan didapati bahwa transformasi dalam ruang parameter $m - c$ adalah sebuah garis lurus dengan persamaan garis dinyatakan sebagai:

$$Yi = mx_i + c \qquad \text{(Pers 2.5)}$$

Sebaliknya jika dalam citra terdapat sebuah garis lurus, dimana garis tersebut mempunyai persamaan seperti persamaan (2.5), maka apabila ditransformasi ke dalam ruang parameter *m – c* akan diperoleh bahwa transformasi dalam ruang parameter *m – c* adalah beberapa garis yang saling berpotongan dalam suatu titik. Persamaan transformasi diperoleh dengan cara memanipulasi persamaan garisnya yaitu persamaan (2.6) menjadi bentuk parametrik menjadi persamaan.

$$c = y_i - mx_i \qquad \text{(Pers 2.6)}$$

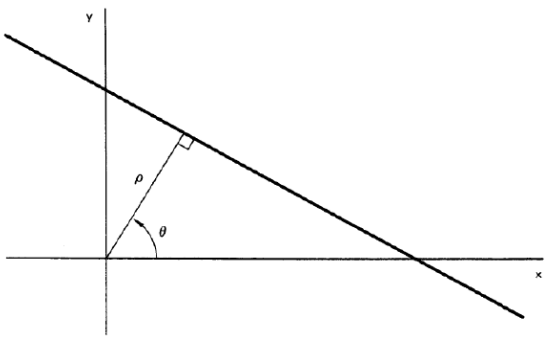
c, m = ruang parameter

x, y = titik pada kurva

Perhitungan transformasi dilakukan dengan cara menghitung tiap titik dalam citra kedalam semua variasi nilai *m - c*.

Dalam kenyataannya, apabila ditemui sebuah garis vertikal, maka akan terjadi masalah dalam penghitungannya dikarenakan garis vertikal mempunyai nilai gradien kemiringan yang besarnya tak berhingga ∞.

Secara khususnya *Hough transform* yang digunakan disini hanyalah untuk mendeteksi garis pada marka jalan saja yang dimana pada awalnya garis jalan ini koordinatnya dalam bentuk *kartesian space (x-y)* yang kemudian dirubah menjadi kurva sinusoidal pada *hough space (rho & theta)* menggunakan *Hough transform* yang dapat dilihat pada Gambar 2.13 dan persamaan (2.7)




**Gambar 2.13** Kartesian Space

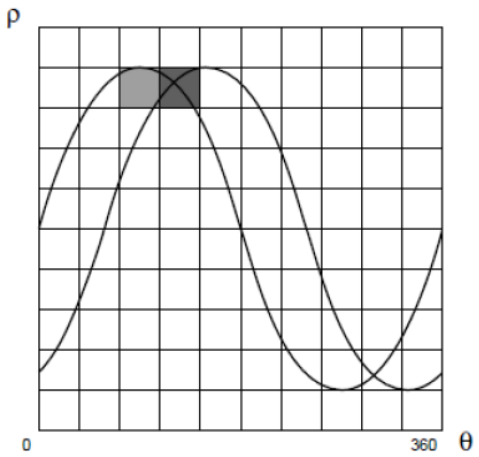
$\rho = x \cos \theta + y \sin \theta$ (Pers 2.7)

$\rho$ = *rho,* koordinat y dalam bentuk *hough space*

$\theta$ = *theta,* koordinat x dalam bentuk *hough space*

Persamaan (2.7) diatas didapatkan dari Gambar 2.13 yang dimana untuk mencari nilai harus menggunakan koordinat x dan y yang telah ditemukan *edge* dan dikalikan dengan nilai yang telah ditemukan juga. Sehingga mendapatkan garis yang diinginkan.

Pada Gambar 2.14 menjelaskan kerja dari *hough transform* dalam melakukan *scanning* pada semua kemungkinan $\theta$ yang ada dan pada pixel yang paling banyak garis sinusoidal yang melewati pixel tersebut merupakan garis yang ditemukan. [**11**]




**Gambar 2.14** Hough *Space*

Kelebihan dan kekurangan dari *Hough transform*:

Kelebihan:

1. HT dapat mendeteksi keberadaan objek yang memiliki pola tertentu walaupun tidak diketahui ukuran maupun posisinya.
2. HT relatif tidak terpengaruh oleh *noise* atau data hilang pada *frame*.
3. Mampu mendeteksi garis pada bentuk geometri dengan kumpulan *edge* yang terputus-putus.

Kekurangan:

1. Kompleksitas dan ukuran *accumulator array* meningkat sebanding dengan bertambahnya jumlah parameter yang merepresentasikan objek yang ingin dideteksi.
2. Biaya komputasi dan kapasitas memori yang besar.
3. Hasil deteksi sensitif terhadap pemilihan nilai *threshold* yang sudah ditentukan, semakin kecil nilai *threshold* maka akan semakin banyak garis yang terdeteksi.

Metode *Hough transform* mempunyai kelebihan yang mampu mendeteksi marka yang putus-putus. Proses pendeteksian tersebut harus mempunyai titik potong pada kurva, maka dapat dimasukkan ke dalam rumus (2.8)

$$\frac{y2 - y1}{x2 - x1} = \frac{y2 - y}{x2 - x} \qquad (2.8)$$

x1 = titik x1 yang diketahui dari kurva

x2 = titik x2 yang diketahui dari kurva

y1 = titik y1 yang diketahui dari kurva

y2 = titik y2 yang diketahui dari kurva

x = nilai titik potong x yang dicari

y = nilai titik potong y yang dicari

## 2.9 Estimasi Posisi Kendaraan Terhadap Marka

Estimasi posisi kendaraan terhadap marka adalah perkiraan posisi kendaraan terhadap marka kiri dan kanan yang dilalui oleh kendaraan itu sendiri. Kendaraan yang dimaksud adalah mobil pribadi yang dipakai untuk berkendara.

Adapun rumus untuk mencari estimasi posisi kendaraan terhadap marka pada persamaan (2.9) dan (2.10). Sedang untuk mengetahui rentangnya pada persamaan (2.11):

$$R1 = (Tengah - B1) - P1 \qquad (2.9)$$

$$R2 = P2 - (Tengah + B2) \qquad (2.10)$$

$$R = R1 + R2 \qquad (2.11)$$

Keterangan:

P = Titik x dari y maximum

B1 & B2 = Batas samping mobil

R1 & R2 = Rentang B ke P

Tengah = Panjang kurva x / 2

Adapun rumus untuk mencari kemana arah kendaraan bergerak marka kiri atau kanan, menggunakan persamaan (2.12):

$$\frac{R1-R2}{R} . 100 \qquad (2.12)$$

Pada persamaan (2.12), nilai dikalikan 100 adalah untuk mencari nilai dalam pentuk persentase.

Berikut adalah *rule* untuk mengetahui kendaraan bergerak ke arah marka kiri dan kanan:

Arah mobil
- R1 > R2, $\frac{R1-R2}{R} . 100$ maka arah kendaraan ke marka sebelah kanan.
- R1 < R2, $\frac{R2-R1}{R} . 100$ maka arah kendaraan ke marka sebelah kiri.

Else, maka nilainya 0. Arah kendaraan berada di tengah.

## 2.10 Penilaian Mengemudi Keluaran

Berdasarkan hasil diskusi dengan pihak Sukses Mandiri penilaian mengemudi dapat diambil pada nilai persentase Dalam Marka, Pindah Lajur dan Pinggir Pembatas, dengan menggunakan tiga predikat: Baik, Cukup Baik, dan Kurang Baik.

Pengemudi dengan predikat "**Baik**": Jika nilai persentase Dalam Marka dalam range 70-100%. Pengemudi dengan predikat "**Cukup Baik**": Jika nilai persentase Dalam Marka dalam range 40-70%. Pengemudi dengan predikat "**Kurang Baik**": Jika nilai persentase Dalam Marka dalam range 0-40%.

Pada kesimpulan predikat mengemudi jika Pindah Lajur berada pada persentase 40-100% maka akan menghasilkan keluaran "Terlalu Sering Pindah Lajur". Jika Pinggir Pembatas menghasilkan persentase pada 50-100% maka keluaran pada kesimpulan yaitu "Terlalu Sering Di Samping Jalan", jika persentase Pinggir Pembatas 5-50% maka keluaran pada kesimpulan, "Kurangi Berada di Pinggir Jalan".

Untuk mengetahui cara menghitung Dalam Marka, Pindah Lajur, dan Pinggir Pembatas menggunakan rumus (2.13)

$$\sum \frac{\text{dalam marka}}{\text{total } \textit{frame}} \times 100\% \qquad (2.13)$$

Begitu pun sama jika ingin menghitung Pindah Lajur dan Pinggir Pembatas menggunakan rumus seperti dalam marka.

Dalam Marka mempunyai beberapa kondisi yaitu ketika kendaraan ketika berada di lajur kiri dan kanan. Pada setiap lajur mempunyai tiga kondisi yang disebut Dalam Marka yaitu Bagian Dalam Kiri Marka, Bagian Dalam Tengah Marka, dan Bagian Dalam Kanan Marka. Untuk menghitung Dalam Marka pada setiap lajur menggunakan rumus (2.14).

$$\sum \frac{\text{dalam kiri marka}}{\text{Dalam Marka}} \times 100\% \qquad (2.14)$$

Begitu pun sama jika ingin menghitung Bagian Dalam Tengah Marka, dan Bagian Dalam Kanan Marka menggunakan rumus seperti Dalam Kiri Marka.

## 2.11 Pengertian C#

Microsoft C# (disebut C sharp) adalah sebuah bahasa pemrograman yang didesain untuk membangun jangkauan aplikasi enterprise yang berjalan di atas *frame*work .NET. Sebuah evolusi Microsoft C dan Microsoft C++, C# sederhana, modern, aman dan Object Oriented.

C# dikenal sebagai visual C# dalam Visual Studio .Net. Dukungan untuk Visual C# termasuk proyek template, desainer, halaman poperti, kode, model

objek dan fitur lain dari lingkungan pengembangan. *Library* untuk pemrograman visual c# adalah .NET *Frame*work. [**12**]

## 2.12 Unified Modelling Language (UML)

*Unified Modelling Language* (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai Diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari tiga notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*) [**12**]

### 2.12.1 Diagram UML

Menggunakan berbagai macam Diagram dengan fungsi masing-masing untuk menggambarkan setiap proses dari sistem berorientasi objek. Berikut meruapakan beberapa Diagram UML diantaranya [**12**]:

a. *Use Case* Diagram

*Use Case* atau Diagram *use case* merupakan pemodelan yang digunakan untuk menggambarkan kelakuan (*behavior*) dari sistem yang akan dibuat. *Use*

*case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *usecase*.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

b. *Activity* Diagram

Diagram aktivitas atau *Activity* Diagram adalah sebuah Diagram yang menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Dalam Diagram aktivitas yang perlu diperhatikan adalah bahwa Diagram aktivitas menggambarkan aktivitas sistem, bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki seuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

c. *Class* Diagram

Diagram kelas atau *class* Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

d. *Sequence* Diagram

Diagram sekuen adalah Diagram yang menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan Diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya Diagram sekuen yang harus digambarkan adalah sebanyak pendefinisian *use case* yang memiliki proses itu sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada Diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka Diagram sekuen yang harus dibuat juga semakin banyak.

## 2.13 Pemograman Berorientasi Objek

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dab perilaku dalam suatu entitas. Model berorientasi objek bermanfaat untuk memahami masalah, komunikasi dengan ahli aplikasi, pemodelan suatu organisasi, meyiapkan dokumentasi serta perancangan program dan basis data. Pertama-tama suatu model analisis dibuat untuk menggambarkan aspek dasar dari domain aplikasi, dimana model tersebut berisi objek yang terdapat dalam domain aplikasi termasuk deskripsi dari keterangan objek dan perilakunya.

Secara spesifik, pengertian berorientai objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karateristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme) dan *inheritance* (pewarisan).

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karateristik utama, yaitu [**12**] :

a. *Encapsulation*

*Encapsulation* (pengkapsulan) merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas dalam bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain kecuali prosedur yang berada dalam objek itu sendiri.

b. *Inheritance*

*Inheritance* (pewarisan) adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi atribut dan metoda dari induknya langsung. Atribut dan metoda dari objek induk diturunkan kepada anak objek, demikian seterusnya. Pendefinisian objek dipergunakan untuk membangun suatu hirarki dari objek turunannya, sehingga tidak perlu membuat atribut dan metoda lagi pada anaknya, karena telah mewarisi sifat induknya.

c. *Polymorphism*

*Polymorphism* (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda. Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.

## 2.14 Open CV

OpenCV (Open Source Computer Vision *Library*) adalah sebuah pustaka perangkat lunak yang ditujukan untuk pengolahan citra dinamis secara real-time, yang dibuat oleh Intel, dan sekarang didukung oleh Willow Garage dan Itseez. Program ini bebas dan berada dalam naungan sumber terbuka dari lisensi BSD.

Pustaka ini merupakan pustaka lintas platform. Program ini didedikasikan sebagaian besar untuk pengolahan citra secara real-time. Jika pustaka ini menemukan pustaka Integrated Performance Primitives dari intel dalam sistem komputer, maka program ini akan menggunakan rutin ini untuk mempercepat proses kerja program ini secara otomatis. [**13**]

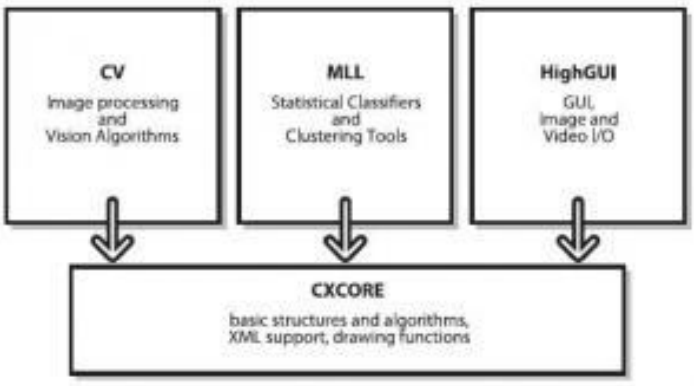
Berikut ini adalah fitur-fitur pada *library* OpenCV:

1. Manipulasi data gambar (alokasi memori, melepaskan memori, kopi gambar, setting serta konversi gambar)
2. *Image*/*Video* I/O (Bisa menggunakan camera yang sudah didukung oleh *library* ini)
3. manipulasi matrix dan vektor serta terdapat juga routines linear algebra (*products, solvers, eigenvalues*, SVD)
4. *Image processing* dasar (filtering, *edge detection*, pendeteksian tepi, sampling dan interpolasi, konversi warna, operasi morfologi, histograms, *image pyramids*)
5. Analisis struktural
6. Kalibrasi kamera
7. Pendeteksian grerak
8. Pengenalan objek
9. Basic GUI (Dis*play* gambar/*video*, *mouse/keyboard* kontrol, *scrollbar*) *Image Labelling* (*line, conic, polygon, text drawing*)

*OpenCV* sendiri terdiri dari 5 *library*, yaitu :

1. CV : untuk algoritma Image processing dan Vision.
2. ML : untuk machine learning *library*
3. Highgui : untuk GUI, Image dan Video I/O.
4. CXCORE : untuk struktur data, support XML dan fungsi-fungsi grafis.
5. CvAux

Berikut adalah struktur dan Konten OpenCV, dapat dilihat pada Gambar 2.15.




**Gambar 2.15 Struktur dan Konten OpenCV**

Open CV yang digunakan di dalam sistem yang akan dibangun yaitu OpenCVSharp yaitu yang berisikan tentang *method image processing*, yang didalamnya mempunyai banyak *method*. Namun yang akan dipakai dan dipanggil dalam sistem antara lain: *grayscale*, *threshold*, filter *smooth*, operasi morphologi dan *hough transform*.

Fungsi pemanggilan *grayscale* pada opencv adalah *cvtColor* dengan bentuk fungsi transformasi dari RGB ke bentuk gray dengan `BgrToGray`, adapun parameter-parameter didalamnya seperti pada Tabel 2.1:

**Tabel 2.1** Parameter cvtColor OpenCV

| Nama Parameter | Deskripsi |
|---|---|
| Src | Source gambar asli hasil inputan berisi nilai piksel RGB 8 bit, 16 bit atau nilai single-presisi |
| dst | Destinasi keluaran gambar dengan ketentuan sama dengan parameter src. |
| code | Kode konversi ruang warna |
| dstCn | Jumlah channel pada gambar tujuan, jumlah chanel berasal secara otomatis dari src dan code. |

Fungsi pemanggilan filter *smooth* pada opencv adalah *Cv.Smooth*, adapun parameter yang ada didalam fungsi *cv.smooth* seperti pada Tabel 2.2:

**Tabel 2.2** Parameter *Cv.Smooth* OpenCV

| Nama Parameter | Deskripsi |
|---|---|
| Src | Source gambar asli hasil inputan. |
| dst | Destinasi keluaran gambar. |
| Smoothtype | Memilih median filter |
| Size1 | Nilai matriks parameter operasi smoothing. Angka yang dimasukkan harus angka ganjil yang positif seperti (1,3,5,7 ...) |

Fungsi pemanggilan *threshold* pada opencv adalah *threshold*, adapun parameter yang ada didalam fungsi *threshold* seperti pada Tabel 2.3:

**Tabel 2.3** Parameter *threshold* OpenCV

| Nama Parameter | Deskripsi |
|---|---|
| Src | Input array (single-chanel, 8 bit atau 32bit) |
| dst | Keluaran array yang mempunyai ukuran yang sama pada parameter src. |
| thresh | Nilai *threshold* |
| maxval | Maksimum nilai yang untuk yang digunakan pada tipe *threshold* |
| Type | Memilih tipe binary |

Fungsi pemanggilan operasi morphologi pada opencv adalah *morphologiEx*, adapun parameter yang ada didalam fungsi *morphologiEx* seperti pada Tabel 2.4:

**Tabel 2.4** Parameter *morphologiEx* OpenCV

| Nama Parameter | Deskripsi |
|---|---|
| Src | Input array (single-channel, 8-bit or 32-bit) |
| Dst | Keluaran array yang mempunyai nilai yang sama dengan parameter src. |
| op | Tipe dari operasi morphologi adalah sebagai berikut: MORPH_OPEN opening<br>MORPH_CLOSE closing<br>MORPH_GRADIENT morphological gradient<br>MORPH_TOPHAT “top hat”<br>MORPH_BLACKHAT “black hat” |
| kernel | Struktur element. |

| | |
|---|---|
| anchor | Posisi jangkar dalam elemen. Nilai default Point (-1, -1) berarti bahwa jangkar adalah di pusat elemen. |
| iterations | Beberapa kali erosi dan dilasi untuk diterapkan |
| steam | Streaming untuk versi asynchronous |

Fungsi pemanggilan *hough transform* pada opencv adalah *HoughLine2*, adapun parameter yang ada didalam fungsi *HoughLine2* seperti pada Tabel 2.5

**Tabel 2.5** Parameter *HoughLine2* OpenCV

| Nama Parameter | Deskripsi |
|---|---|
| dst | Keluaran gambar yang akan terdeteksi oleh garis |
| lines | Sebuah vektor yang akan menyimpan parameter ($\rho$,$\theta$) dari deteksi garis |
| rho | Resolusi dari parameter $\rho$ dalam pixel. |
| theta | Resolusi dari parameter $\theta$ dalam radian. |
| *threshold* | Angka minimum dari interseksi untuk mendeteksi sebuah garis |
| srn and stn | Nilai parameter pendukung dari garis |