BAB 2

LANDASAN TEORI

2.1. Teori Umum

2.1.1. Warna

Dengan menggunakan 3 buah reseptor manusia dapat membedakan banyak warna. Warna *tricromatic* RGB dalam sistem grafis umumnya menggunakan 3 byte $(2^8)^3$, atau sekitar 16 juta kode warna. Dikatakan kode warna dan bukan warna karena manusia tidak dapat membedakan warna sebanyak itu. Mesin dapat membedakan antara berbagai kode warna, namun perbedaan tersebut belum tentu dapat menunjukan perbedaan yang dapat ditangkap oleh mata manusia. Tiap 3 byte pada pixel RGB mencangkup 1 byte pixel untuk tiap warna *Red* (merah), *Green* (hijau), *Blue* (biru) (Saphiro & Stockman, 2001).

Gambar direpresentasikan sebagai matrik dengan elemen *integer* atau *pixel*. Biasanya *pixel* tidak disimpan dalam bentuk matrik sederhana, namun menggunakan data yang lebih rumit. Dan terkadang memudahkan secara matematis jika kita beranggapan gambar berwarna sebagai matrik dari vektor tiga dimensi(Pavlidis, 1982).

$$Y = (0.299)R + (0.587)G + (0.114)B$$

Persamaan 2-1 Rumus perubahan nilai *pixel* RGB menjadi *grayscale* pada *OpenCV*.

Bila dibandingkan dengan gambar biasa, gambar *grayscale* memilki akurasi yang kurang, hal ini terutama dikarenakan warna dapat membantu kita dalam membedakan objek dengan lebih baik. Namun untuk beberapa aplikasi penggunaan warna akan meningkatkan pengeluaran, sehingga tidak diusulkan. Hal ini dikarenakan, tidak hanya karena kamera berwarna yang harganya lebih mahal, juga disebabkan karena dengan menggunakan warna maka diperlukan tiga buah *digitizer*, dan tiga buah *frame* untuk memasukkan gambar tersebut. Dan karena dengan menggunakan gambar berwarna maka diperlukan pemrosesan yang lebih banyak dalam menginterpretasikan gambar berwarna dengan sepenuhnya (Davies, 1990).

Sedangkan pada *grayscale*, tiap warna direpresentasikan oleh 1 bit pixel, dimana tiap bit dari *pixel* yang merepresentasikan gambar *grayscale* adalah suatu nilai yang menunjukkan nilai intensitas dari gambar yang berada pada posisi tersebut. Sehingga dapat diproses sebagai entitas tunggal pada komputer. Tidak seperti gambar RGB, yang merupakan suatu set yang berisi tiga bit data, maka diperlukan tiga buah angka untuk merepresentasikan informasi yang ada didalamnya. Sehingga pada gambar *grayscale* diperlukan proses komputasi yang lebih sedikitdi bila bandingkan dengan gambar berwarna.

2.2. Teori Khusus

2.2.1. Smoothing

Smoothing merupakan salah satu teknik yang umum digunakan pada pengolahan gambar. Proses ini bertujuan untuk memperhalus gambar, mengurangi resolusi gambar, serta mengurangi *noise* (Bradski & Kaehler, 2008).

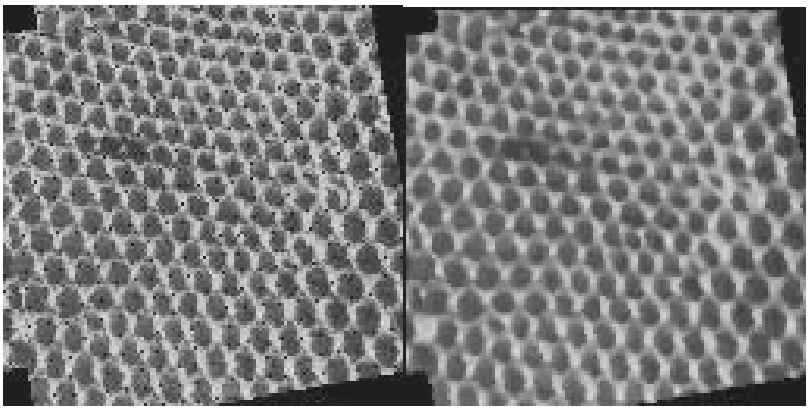
Pada prosesnya *smoothing* menggunakan teknik konvolusi yang menggunakan kernel dengan berbagai ukuran. Pada pengerjaanya terdapat beberapa cara untuk melakukan *smoothing*, seperti *mean filter*, *median filter*, *gaussian filter*, dan *bilateral filter*. Dimana pada proses paling sederhananya hasil dari *smoothing* pada suatu *pixel* adalah hasil dari rata-rata *pixel* tersebut dengan *pixel* dekitarnya,

2.2.1.1. Median Filter

Konsep dasarnya adalah dengan menemukan nilai *pixel* yang memiliki nilai intensitas dari suatu *pixel* yang berbeda dengan nilai *pixel* yang ada di daerah 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 (Davies, 1990).

Untuk itu kita perlu untuk 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.

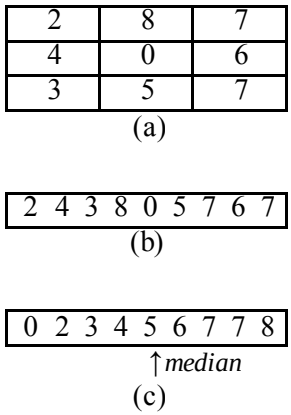


(a) (b)

Gambar 2-1 Proses *median filter*,
(a) gambar awal; (b) gambar hasil *median filter* dimana terlihat lebih halus dan *noise* hilang.

Berbeda dengan *gaussian filter* yang menghaluskan keseluruhan gambar, pada *median filter* terlihat bahwa proses penghalusannya terjadi pada daerah tepi gambar. Sehingga meski terjadi penghalusan gambar, *median filter* lebih kearah "melembutkan" gambar yang ada.

Kita dapat pula mengurangi komputasi dengan mencari nilai extrem yang perlu untuk diubah, atau dengan membatasi nilai *median* yang akan digunakan.



Gambar 2-2 Mencari *median* dengan ukuran 3x3.
(a) bentuk nilai intensitas yang ditemukan; (b) penempatan nilai intensitas dalam bentuk vektor; (c) nilai intensitas yang telah diurutkan, dan ditemukan *median*-nya.

Terlihat diatas bahwa pengurutan nilai intensitas *pixel* wajib untuk digunakan. Bila kita menggunakan *buble sort* maka untuk setiap daerah n x n, maka diperlukan fungsi sebanyak O (n^4). Maka untuk *median filter* yang memiliki daerah kecil, antara sebesar 3x3 atau 4x4, maka *buble sort* atau penyortiran lain baik untuk digunakan. Namun tidak terlalu baik bila daerah yang dipakai lebih dari atau sama dengan 5x5, atau nilai inensitasnya lebih terbatas.

Median filter umumnya menggunakan kernel dengan ukuran 3x3. Namun dapat pula menggunakan ukuran yang lebih besar. Selain itu sesuai dengan perkembanganya maka bentuk yang

dipakai juga dapat bemacam-macam, seperti salib, dan garis (vertikal dan horisontal), yang terpusat pada titik tengahnya. Hal ini dimaksudkan agar proses yang dihasilkan menjadi lebih cepat, terutama karena jumlah *pixel* yang dihitung menjadi lebih sedikit. *median filter* cukup dikenal baik atas kemampuanya untuk menghilangkan *salt and paper noise*. Selain itu *median filter* akan meningkatkan kualitas gambar, sehingga memperjelas daerah tepi (*edge*) pada gambar (Nixon & Aguado, 2002).

Hal ini terjadi karena pada daerah yang terletak pada bagian tepi suatu gambar, *filter* akan memproses data dan umumnya akan mendapatkan nilai yang sesuai dengan daerah yang memiliki nilai intensitas yang lebih besar(didalam atau dilar batas). Sehingga *filter* secara tidak langsung menentukan terdapat pada bagian mana *pixel* itu berada. Hal ini tentu saja membuat daerah tepi menjadi sedikit melebar, namun perlu diingat bahwa *pixel* melebarkan daerah tepi dari kedua belah sisi, sehingga hal ini menyebabkan daerah tepinya bisa lebih terlihat (Davies, 1990).

2.2.1.2. Gaussian Filter

Gaussian filter merupakan salah satu proses memperhalus gambar yang dinilai sudah cukup optimal. Kernel yang digunakan pada *gaussian filter* telah ditetapkan dengan menggunakan fungsi *Gaussian relationship*. Dimana nilai fungsi *gaussian* pada $\boldsymbol{g}$ dengan

koordinat *x,y* diatur oleh variabel σ^2 sesuai dengan dimensi yang dihitungnya (Nixon & Aguado, 2002).

Untuk perhitungan 1D, maka :

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

Persamaan 2-2 *Gaussian relationship* 1D.

Sedangkan pada gambar 2D :

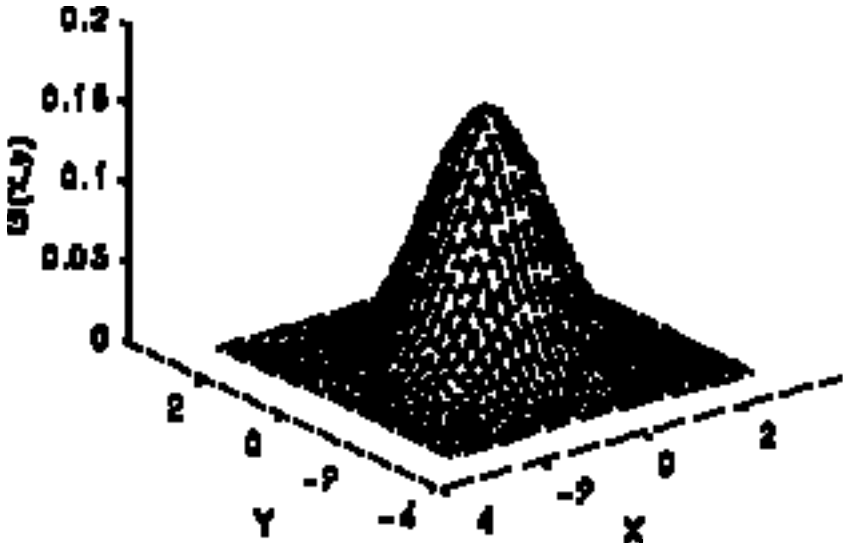
$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

atau

$$g(x,y) = e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

Persamaan 2-3 *Gaussian relationship* 2D.

Sehingga memiliki distribusi yang *menyerupai* kerucut seperti yang terlihat pada Gambar 2-3.



Gambar 2-3 Distibusi *gaussian* 2D.

Dimana x adalah jarak dari titik asal pada sumbu horisontal, y adalah jarak dari titik asal pada sumbu vertikal, dan σ adalah standar derivasi dari distribusi *gaussian*.

Pada Gambar 2-4 , terlihat bahwa pada bagian pusatnya terdapat titik yang menonjol. Titik inilah yang nantinya akan digunakan sebagai pusat. Sehingga akan didapat kernel yang akan digunakan dalam konvolusi. Nilai bagian pusat kernel memiliki nilai tertinggi, sedangkan nilai pada bagian lainnya memiliki nilai yang lebih kecil, dan semakin menjauhi pusat maka nilainya akan semakin mengecil.

0.02	0.08	0.14	0.08	0.02
0.08	0.37	0.61	0.37	0.08
0.14	0.61	1.0	0.61	0.14
0.08	0.37	0.61	0.37	0.08
0.02	0.08	0.14	0.08	0.02

Gambar 2-4 Kernel *gaussian blur* 5x5 ($\sigma = 1$)

Meski kernel *gaussian filter* memiliki berbagai ukuran (3x3, 5x5, 7x7), namun semakin besar ukurannya gambar yang dihasilkan cenderung menurunkan kualitas gambar yang dihasilkan. Hal ini terlihat pada Gambar 2-5.

(a) (b) (c)

Gambar 2-5 *Gaussian filter,*

(a) *Gaussian filter* 3x3; (b) *Gaussian filter* 5x5; (c) *Gaussian filter* 7x7.

2.2.2. Morfologi

Nama morfologi secara matematis didapatkan dari pembelajaran tentang bentuk. Pendekatan ini menggali fakta bahwa di dalam banyak aplikasi bahasa mesin adalah wajar, dan mudah untuk berfikir dalam bentuk saat merancang algoritma. Pendekatan morfologikal memfasilitasi

pemikiran berbasiskan bentuk atau ikonis. Dalam pendekatan morfologi, unit yang paling dasar dari informasi bergambar adalah gambar biner (Jain, Katsuri, & Schunck, 1995).

Morfologi matematis mendeskripsikan suatu aplikasi operator aljabar, yang dapat digunakan untuk mengambil suatu daerah, bentuk dan batas pada gambar(Kulkarni, 2001).

2.2.2.1. Dilatasi

Dilatasi merupakan proses menambahkan atau mempertebal objek pada gambar biner(Zhou, Wu, & Zhang, 2010). Hal ini dilakukan dengan memberikan tambahan *pixel* pada batasan suatu objek pada suatu gambar. Jumlah *pixel* yang ditambahkan tergantung dari ukuran dan bentuk dari *stucturing element* yang digunakan pada pengolahan gambar. Dimana *stucturing element,* merupakan sebuah matrik yang hanya berisi 1 dan 0, yang dapat memiliki berbagai bentuk dan ukuran.

Secara matematis dilatasi A oleh B, dinotasikan $A \oplus B$, didefinisikan sebagai Persamaan 2-4.

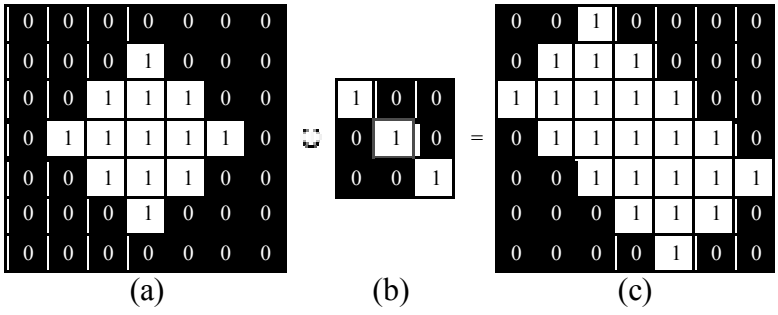
$$A \oplus B = \left\{ z \middle| (\hat{B})_z \cap A \neq \emptyset \right\}$$

Persamaan 2-4 Definisi matematis dilatasi.

Dimana $\emptyset$ merupakan set yang kosong dan B adalah *stucturing element*-nya. Atau dapat juga dikatakan sebagai dilatasi A oleh B merupakan suatu set yang didalamnya terdiri dari semua

stucturing element dari lokasi awalnya, dimana B yang telah direfleksikan, dan ditranslasikan akan tumpang tindih setidaknya dengan sebagian dari gambar A.

Secara algoritma operasi ini dapat didefinisikan sebagai : Anggaplah bahwa B merupakan sebuah *mask*. Kemudian taruhlah titik referensi dari *stucturing element* pada *pixel* yang memiliki nilai 1 pada gambar A. dan kemudian lakukan fungsi *OR* pada A dan B. Sehingga bila terdapat nilai yang berbeda pada *pixel* A dan B yang memiliki lokasi yang sama, maka nilai dari keluarannya adalah 1. Namun bila bilai pada *pixel* A dan B sama maka nilai keluarannya akan sama dengan nilai terebut. Proses ini kemudian akan diulang terus menerus, hingga nilai titik referensi B telah berada di semua titik A yang memiliki nilai 1. Untuk lebih jelasnya dapat dilihat pada Gambar 2-7.



Gambar 2-7 Contoh proses dilatasi,

(a) merupakan gambar awal, dengan objek bernilai 1; (b) structuring element dengan titik referensi yang diberi kotak merah; (c) hasil akhir proses dilatasi.

Dilatasi pada umumnya memperbesar ukuran suatu objek, menutupi lubang, dan menghubungkan *area-area* yang lebih kecil dari struktur elemennya.

2.2.2.2. Erosi

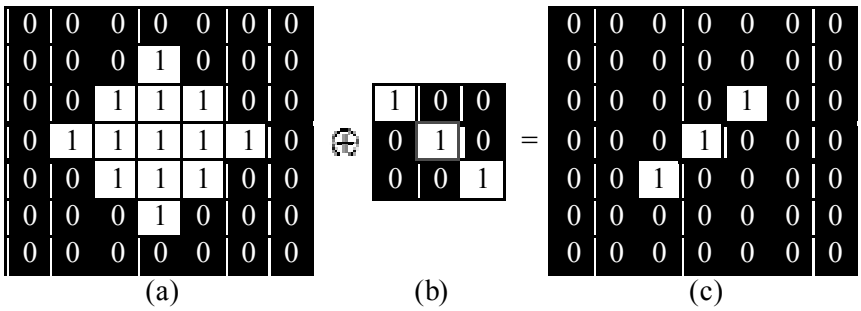
Erosi merupakan proses untuk mengecilkan ataupun menipiskan objek pada gambar(Zhou, Wu, & Zhang, 2010). Sama seperti dilatasi, proses ini juga dipengruhi oleh *structuring element* yang digunakan pada pengolahan gambar.

Definisi matematis erosi mirip dengan dilatasi. Dimana erosi A oleh B, dinotasikan $A \ominus B$, didefinisikan sebagai Persamaan 2-5.

$$A \ominus B = \{x | (B)_x \cap A^c \neq \phi\}$$

Persamaan 2-5 Definisi matematis erosi.

Secara algoritma kita dapat mendefinisikan erosi sebagai : Pertama-tama tiap nilai yang ada pada keluaran $A \ominus B$ akan dijadikan nilai 0 (*zero*). B kemudian akan ditempatkan pada tiap lokasi A yang memiliki nilai *pixel* 0 (dengan mengacu pada titik referensi B). Bila pada A terdapat B (pada proses A *AND* B tidak sama dengan *zero*) maka pada keluaran akan digunakan nilai B. Hasil keluaran akhir adalah setiap elemen dari A yang tidak terkena elemen B pada proses diatas. Untuk lebih jelasnya dapat dilihat pada Gambar 2-8.



Gambar 2-8 Contoh proses erosi,

(a) merupakan gambar awal, dengan objek bernilai 1; (b) *structuring element* dengan titik referensi yang diberi kotak merah; (c) hasil akhir proses erosi.

2.2.2.3. Opening dan *Closing*

Pada prakteknya penggunaan erosi dan dilatasi sering dikombinasikan satu dengan yang lainnya dalam memproses suatu gambar. Suatu gambar dapat diproses oleh dilatasi atau erosi ataupun keduanya yang terjadi secara terus menerus, baik dengan *structur element* yang sama. Operasi yang paling umum digunakan dikenal dengan nama *opening* dan *closing*.

2.2.2.3.1. *Opening*

Opening merupakan suatu proses dimana suatu gambar yang terlebih dahulu mengalami proses erosi, disusul dengan proses dilatasi dengan menggunakan struktur elemen yang sama. Hal ini bertujuan untuk memulihkan sebesar mungkin data yang telah hilang karena erosi pada suatu gambar. Selain itu erosi juga dapat digunakan untuk menghilangkan objek yang terlalu kecil, memisahkan antar objek yang jaraknya berdekatan.

Opening A oleh B, yang didenotasikan dengan $A \circ B$ didefinisikan sebagai :

$$A \circ B = (A \ominus B) \oplus B$$

Persamaan 2-6 *Opening*.

Sehingga $A \circ B$, merupakan gabungan dari semua translasi *B* yang termasuk dalam *A*. *Opening* membuang semua bagian dari objek yang tidak dapat menampung *structuring element*-nya. Sehingga memperhalus bagian tepi objek tanpa merusak bentuk aslinya, dan mengilangkan objek yang terlalu kecil atau tipis.

(a) (b) (c)

Gambar 2-9 Contoh proses *opening*,
(a) gambar asli; (b) gambar asli yang telah mengalami erosi; (c) gambar (b) yang telah didilatasi dengan structuring element yang sama.

2.2.2.3.2. *Closing*

Closing merupakan kebalikan dari *opening*, dimana suatu gambar mengalami proses dilatasi terlebih dahulu sebelum kemudian dierosi dengan struktur elemen yang sama.

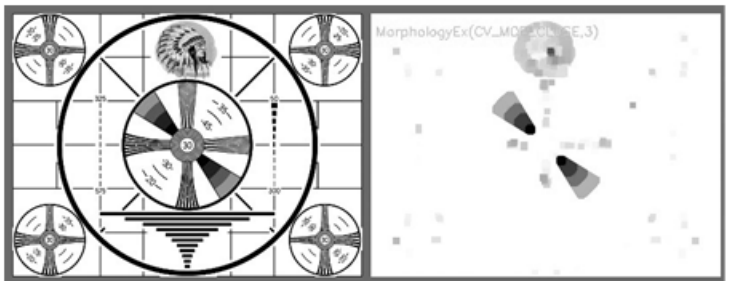
Operasi ini biasanya digunakan untuk menghilangkan *noise* yang ada pada gambar.

Operasi *closing* A oleh B, yang didenotasi sebagai $A \cdot B$ adalah:

$$A \cdot B = (A \oplus B) \ominus B$$

Persamaan 2-7 *Closing.*

Closing $A \cdot B$ merupakan komplemen dari semua gabungan dari semua translasi *B* yang tidak tumpang tindih dengan *A*. Hal ini memungkinkan operasi ini untuk menutup lubang kecil atau garis tipis pada suatu objek, dan menghubungkan 2 objek yang letaknya berdekatan, serta menghaluskan batas tiap objek tanpa merusak bentuk aslinya.



Gambar 2-10 Contoh proses *closing*,
dimana garis-garis hitam (latar) menjadi lebih sedikit, sedangkan bagian putih (objek) membesar.

2.2.3. *Threshold*

Masalah utama yang ada pada aplikasi yang memakai *computer vision* adalah pembedaan subgambar yang merepresentasikan objek.

Operasi yang mudah dilakukan oleh mata ini sulit dilakukan oleh komputer (Jain, Katsuri, & Schunck, 1995). Sehingga dalam proses pengelompokan suatu gambar, biasanya berdasarkan intensitas, warna, dan faktor lainnya. Sehingga *thresholding* dapat digunakan untuk memisahkan gambar dari latarnya.

Input yang dimasukkan dalam *thresholding* adalah gambar *grayscale*. Sedangkan hasil keluarannya umumnya berbentuk gambar biner yang merepresentasikan hasil dari segmentasi tersebut (Davies, 1990).

Pada teknik ini kecerahan warna dari setiap *pixel* dibandingkan kedalam sebuah nilai *threshold*, dan *pixel* hasilnya dinyatakan sebagai satu atau dua kategori, putih atau hitam, tergantung dari apakah nilai pixel aslinya melampaui atau tidak nilai *threshold*-nya. Seleksi dari angka *threshold* biasanya didapat dari histogram. Jika sebuah gambar terdiri dari dua *area*, satu adalah daerah terang utama yang terang dan yang lain adalah daerah gelap, maka kita bisa mengasumsikan sebuah histogram yang mempunyai dua puncak. Dan nilai *threshold* bisa dipilih berdasarkan nilai antara dua puncak pada histogramnya.

Thresholding adalah teknik yang paling sederhana yang dapat digunakan untuk segmentasi. Sayangnya tidak selalu mungkin untuk menentukan nilai *threshold* dikarenakan rata-rata tingkat kecerahan bisa

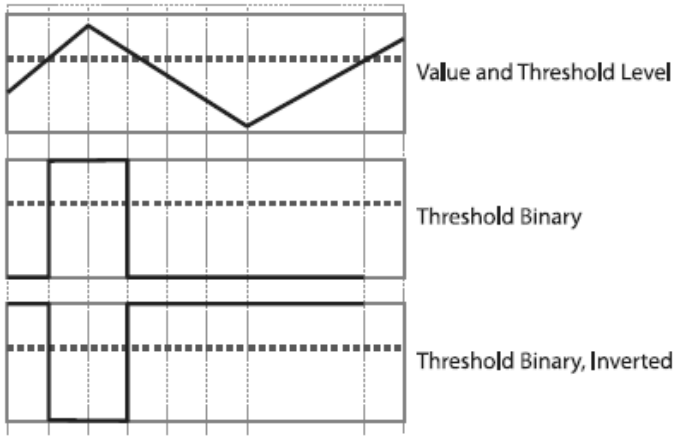
saja berbeda dan kita harus mengevaluasi histogram dari setiap gambar(Pavlidis, 1982).

Pada saat diimplementasikan, maka segmentasi akan berlangsung sesuai dengan nilai intensitas yang diberikan. Sehingga bila nilai intensitas suatu *pixel* lebih dari nilai intensitas yang diberikan maka warna *pixel* tersebut akan menjadi hitam. Begitu pula kebalikannya, bila nilai intensitas suatu *pixel* kurang dari atau sama dengan nilai intensitas yang diberikan maka *pixel* tersebut akan diberi warna putih. Atau dapat pula dilakukan beberapa *threshold*, sehingga kita dapat memberikan batas-batas untuk *area* intensitas tertentu yang ingin disegmentasikan. *Threshold* dapat didefiniskan seperti pada Persamaan 2-8.

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{if } f(x,y) < T \end{cases}$$

Persamaan 2-8 *Thresholding.*

Ada pula teknik *threshold* yang tidak hanya menghasilkan gambar hitam dan putih melainkan gambar berwarna. Pada teknik ini umumnya *pixel* yang seharusnya berwarna putih diganti dengan nilai *pixel* asli. Keunggulan utama dari teknik ini adalah terjaganya warna, sehingga tidak banyak informasi yang terbuang.



Gambar 2-11 Skema *threshold*,

dimana garis putus-putus menandakan nilai intensitas yang diberikan.

Penentuan nilai intensitas merupakan faktor yang penting dalam penentuan hasil dari *thresholding*. Salah satu cara untuk memperbaiki kualitas *thresholding* adalah dengan melakukan *background subtraction* sebelum melakukan *threshold*. Hal ini dapat membantu dalam proses pengolahan gambar, terutama bila *background* yang ada cukup statis. Namun meski cara ini cepat dan mudah untuk diimplementasikan. Dalam penggunaannya, metode ini dapat menimbulkan *noise* yang tidak diinginkan, dan sensitif terhadap cahaya(Nixon & Aguado, 2002).

2.2.4. *Hough Transform*

Hough Transform merupakan teknik untuk mencari bentuk didalam suatu gambar (Nixon & Aguado, 2002). Keunggulan utama dari metode ini adalah karena dengan menggunakan metode ini maka dapat diperoleh hasil yang sama dengan metode mencocokan template, namun dengan hasil yang lebih cepat. Hal ini dimungkinkan dengan mereformulasi proses dari mencocokan template.

Pada implementasinya *Hough Transform* mendefinisikan pemetaan dari titik-titik pada gambar kedalam *area* akumulasi (*hough space*). Pemataan ini terjadi dalam proses komputasi yang efisien, bedasarkan fungsi yang menggambarkan bentuk dari targetnya. Pemetaan ini membutuhkan lebih sedikit sumber daya dalam komputasinya bila dibandingkan dengan mencocokan template. Namun masih membutuhkan kapasitas penyimpanan dan komputasi yang cukup tinggi dalam penggunaannya. Tetapi, fakta bahwa *Hough Transform* setara dengan pencocokan template telah memberikan dorongan yang cukup untuk menjadikannya teknik ekstraksi bentuk yang cukup populer.

2.2.4.1. Hough circle

Awalnya metode ini dilakukan dengan memperkirakan intensitas dari tiap gradien yang ada pada gambar. Kemudian gambar akan mengalami *threshold* untuk menghasilkan tepian yang signifikan. Dilanjutkan menentukan tiap titik yang dimungkinkan sebagai titik pusat lingkaran. Dimana tiap titik tersebut beserta jari-jarinya akan disimpan kedalam *parameter space*. Kemudian dicari nilai fluktuasi yang berhubungan dengan titik pusat yang yang ada(Davies, 1990).

Hough Transform dapat mencari lingkaran dengan mengganti persamaan atas kurva dalam proses deteksi. Persamaan ini bisa dalam bentuk eksplisit, ataupun parametrik. Dalam bentuk eksplisit,

Hough Transform dapat didefinisikan setelah mempertimbangkan Persamaan 2-9 (Nixon & Aguado, 2002).

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

Persamaan 2-9 Persamaan eksplisit lingkaran.

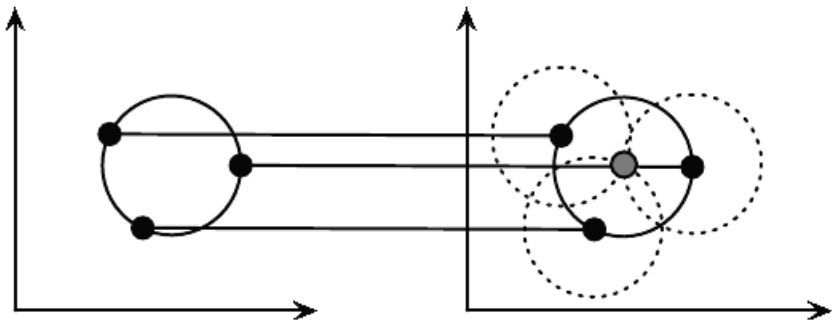
Persamaan ini mendefinisikan kedudukan titik (*x*,*y*) dari titik pusat (x_0,y_0), dengan jari-jari *r*. atau mendefinisikan kedudukan titik (x_0,y_0) dari titik pusat (*x*,*y*), dengan jari-jari *r*.

Sedangkan Persamaan 2-9 dapat dijadikan persamaan parametrik (Persamaan 2-10).

$$x = x_0 + r\cos\theta \qquad y = y_0 + r\sin\theta$$

Persamaan 2-10 Persamaan parametrik lingkaran.

Pada pengerjaannya *hough circle* akan mencari titik-titik yang diduga sebagai bagian tepi dari suatu lingkaran. Kemudian dengan menggunakan persamaaan diatas akan mencari nilai *r*, x_0, y_0. Hal ini bertujuan untuk menemukan titik pusat sebenarnya dari lingkaran yang mengandung titik-titik tersebut. Titik pusat sebenarnya baru akan diambil bila terdapat perpotongan antara lingkaran yang dibuat oleh *hough circle* (Gambar 2-13).



Gambar 2-13 Mendeteksi titik pusat pada *hough circle*.

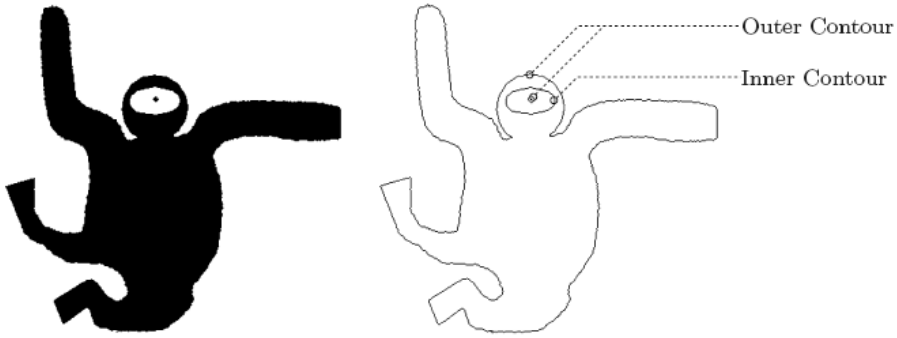
Bila terdapat banyak lingkaran pada suatu gambar maka akan ada kemungkinan timbulnya suatu titik pusat yang tidak sesuai. Untuk itu dapat dilakukan perbandingan dengan gambar aslinya.

2.2.5. *Contour Finding*

Salah satu masalah yang paling umum pada analisis grafis dan gambar adalah menemukan interior dari sebuah *area* dimana kontur diberikan (Pavlidis, 1982). Masalah ini dapat diselesaikan dalam beberapa cara, dimana bisa dibagi menjadi dua kelas besar. Yang pertama, hasil mempunyai deskripsi dari kontur sebagai *polygon* dan menentukan bagian mana dari bidang yang terletak pada interoir dengan mempertimbangkan efek persamaan garis. Teknik tersebut biasanya dikatakan berbasis *polygon*. Metode dari kelas kedua, memetakan kontur kedalam bidang yang berlainan dan kemudian menentukan lokasi dari interior dengan mengamati nilai dari *pixel*.

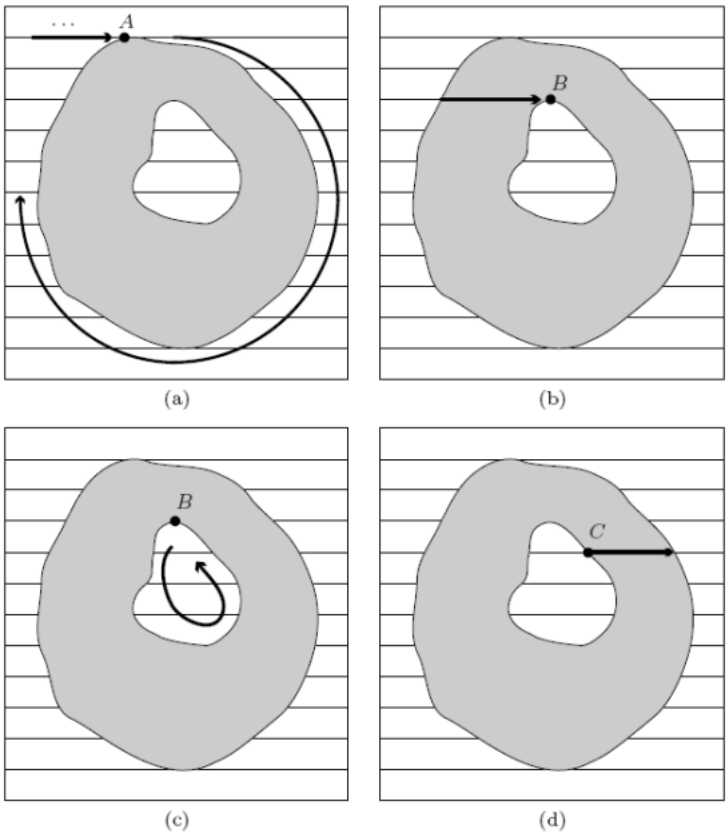
Suatu bagian gambar yang berhubungan hanya akan memiliki satu kontur luar, namun dapat memiliki beberapa kontur dalam akibat dari adanya “lubang”. Sehingga pada lubang tersebut masih dapat memiliki

beberapa daerah lain didalamnya. Selain itu ada pula daerah yang terhubung dengan satu pixel yang sama. Sehingga dapat terjadi kesalahan dalam penentuan titik akhir suatu kontur.



Gambar 2-6 Gambar biner yang memiliki kontur luar dan dalam.

Hal ini dapat dihindari dengan melakukan penggambungan antar konsep pemberian label daerah secara sekuensial, dengan konsep penentuan kontur kedalam sebuah algoritma. Hal ini dilakukan dengan pemberian label pada daerah gambar dan pada saat yang bersamaan melakukan penentuan kontur luar dan dalam (Burger & Burger, 2009).



Gambar 2-7 Proses kerja *contour finding*,
(a) ditemukan titik awal, dan penelusuran garis yang berhubungan; (b) ditemukannya titik awal kontur dalam; (c) penelusuran kontur awal; (d) penentuan bagian interor.

Gambar 2-6, memperlihatkan cara pengerjaan dari proses *contour finding* menggunakan metode ini, dimana :

1. Pertama kali akan terdapat pengecekan keseluruhan pixel dari bagian paling kiri atas hingga bagian paling kanan bawah.
2. Pada saat melakukan pengecekan terdapat beberapa kasus yang mungkin terjadi :

   Kasus A : Transisi dari suatu pixel latar, menuju pixel latar lainnya (Gambar 2-6 (a)). Ini menunjukkan bahwa pixel

ini berada pada kontur luar suatu wilayah baru. Sebuah label baru kemudian akan diberikan dan garis yang berhubungan akan dilalui dan ditandai. Dan daerah *background* yang berhubungan langsung dengan daerah tepi luar akan diberi label kusus -1.

Kasus B : Bila terjadi transisi antara daerah dalam (*interior*) dengan daerah *background* yang belum ditandai (Gambar 2-6 (b)) maka daerah tersebut termasuk kontur dalam. Dari titik ini maka semua titik yang merupakan kontur dalam akan dilalui, dan diberi tanda. Dan semua latar yang berhubungan akan diberi label khusus -1.

Kasus C : Ketika sebuah pixel *interior* tidak terdapat pada sebuah kontur, maka pixel disebelah kirinya sudah diberi label (Gambar 2-6 (d)) dan label ini diberikan pula pada pixel ini.

2.2.6. *Background Averaging*

Jika terdapat beberapa gambar pergerakan sebuah objek yang berjalan secara berurutan, dan kita ingin menemukan latar (jadi kita bisa memisahkan subjek yang berjalan dari latar). Kita dapat merata-ratakan gambar untuk dapat mencari latar gambar tersebut (Nixon & Aguado, 2002).

Jika kita melakukan *temporal average*, sebuah gambar dimana setiap titik adalah rata-rata dari titik dari masing – masing gambar yang digunakan, maka kita akan memperoleh hasil dimana objek yang berjalan tampil pada latar, tetapi dengan sangat samar. Objek tersebut menjadi samar karena nilai *pixel* pada objek menjadi berkurang. Misalkan kita memiliki 6 buah gambar pergerakan objek yang berurutan, maka dengan melakukan *temporal average*, maka nilai *pixel* objek akan menjadi 1/6 dari nilai *pixel* aslinya. Pengurangan nilai ini terjadi karena pencarian nilai rata-rata seluruh gambar (1/6 karena gambar yang dimiliki berjumlah 6)

Kita tentu saja dapat menggunakan gambar lebih banyak rata-rata latar tersebut. Sehingga akan mengakibatkan bayangan yang dihasilkan terlihat lebih tipis. Kita juga dapat melakukan *spatial averaging* untuk lebih mengurangi efek dari subjek berjalan. Yang akan menghasilkan *spatio temporal averaging*. Pada kasus ini, kita bisa mengurangi penggunaan gambar yang dijadikan sample tetapi akan menghasilkan latar yang kurang rinci.

2.2.7. *Background Subtraction*

Karena kemudahannya, dan karena pada umumnya lokasi kamera yang digunakan bersifat tetap, maka *background subtraction* merupakan salah satu teknik pemrosesan gambar yang paling dasar, terutama pada aplikasi keamanan yang menggunakan kamera (Bradski & Kaehler, 2008).

Untuk melakukan *background subtraction* maka kita harus "mempelajari" contoh latar. Setelah dipelajari maka latar akan dibandingkan dengan gambar pada *real-time*, kemudian bagian yang masih tergolong latar akan dihilangkan. Hal ini akan menyisakan benda yang dianggap sebagai objek yang berada diatas latar.

Dalam hal-hal tertentu "latar" dapat memiliki definisi yang berbeda-beda untuk tiap aplikasi. Sebagai contoh, bila kita memperhatikan jalan raya maka lalu lintas yang normal mungkin dapat dikatkan sebagai latar.

Namun pada umumnya, latar merupakan segala hal-hal yang sifatnya statis, atau dapat terjadi secara berkala yang ada pada gambar, yang akan selalu statis, atau mengulangi hal yang sama selama waktu yang diinginkan berlangsung. Namun dapat juga memiliki komponen yang dapat berubah seiring waktu, seperti pohon yang bergoyang pada pagi hari, dan diam pada sore harinya.

Terdapat dua hal umum yang membedakan kategori dari lingkungan yang kita pakai, dan sering dijumpai, yaitu lingkungan dalam ruangan, dan lingkungan luar ruangan.