BAB 2
LANDASAN TEORI

2.1 Umum

Saat ini penggunaan kamera dalam bidang industri telah banyak digunakan untuk membantu pekerjaan manusia, salah satunya adalah pengendalian robot industri. Agar kamera dapat terintegrasi dengan lengan robot dengan baik, maka diperlukan suatu sistem yang baik pula. Sistem adalah kumpulan dari entitas seperti manusia atau mesin dan sebagainya, yang bertindak dan berinteraksi secara bersama-sama untuk mendapatkan suatu penyelesaian (Schmidt dan Taylor, 1970, p3). Selain itu menurut Jogiyanto (1989, p813) sistem dapat juga didefinisikan sebagai suatu kesatuan yang terdiri dari 2 atau lebih komponen atau subsistem yang berinteraksi untuk mencapai suatu tujuan. Sedangkan definisi lain dari sistem digagas oleh Mcleod (1995, p33) yaitu sistem adalah sekelompok elemen-elemen yang saling terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan.

Dasar dari pengendalian robot adalah robotnya itu sendiri. Definisi umum dari robot adalah mesin yang dapat diprogram untuk melaksanakan tugas-tugas mekanik. Robot yang berintelegensi dapat memberi respon terhadap perubahan lingkungan (HME-ITB) dan robot adalah sebuah manipulator multifungsi yang dapat diprogram ulang dan didesain untuk memindahkan material, komponen, peralatan melalui variabel-variabel yang ada didalam sebuah program untuk melakukan beberapa tugas (Fairhurst, 1988, p2). Selain itu, Fu (1987, p1) mengatakan robot adalah suatu alat yang dapat diprogram ulang (*reprogrammable*) untuk dapat mengerjakan tugas tertentu yang diinginkan penggunanya. Sedangkan menurut standar ISO sendiri, robot adalah sebuah alat kendali otomatis, dapat diprogram ulang, *multipurpose*, memiliki 3 atau lebih *axis*, baik yang tidak dapat berpindah (*fixed*) maupun yang dapat berpindah (*mobile*) yang digunakan untuk tujuan aplikasi industri otomasi (*International Standard Organization* (ISO 8373)).

Pengendalian robot industri menggunakan kamera berarti pengendalian robot dengan cara mengolah citra yang dihasilkan oleh kamera. Citra adalah representasi informasi 2 dimensi yang diciptakan atau dibuat dengan melihat atau lebih tepatnya merasakan sebuah gambar atau pemandangan (Tharom, 2003). Citra dapat 1 warna (*monochrome*) atau berwarna (Pakpahan, 1984, p13). Dalam pengembangannya ada ilmu komputer yang khusus membahas mengenai pengolahan citra, yaitu *Computer Vision*. *Computer vision* membuat komputer memiliki kemampuan untuk mengenali benda yang dilihatnya, berupa sebuah gambar atau adegan nyata dengan cara mengidentifikasi obyek, ciri atau polanya (*pattern*).

Kamera sebagai alat *input* dalam pengendalian robot bertugas untuk menangkap gambar yang "dilihatnya". Kamera sendiri didefinisikan sebagai sebuah peralatan untuk mengambil foto (gambar), gambar bergerak, atau gambar televisi. Kamera dalam arti umum sendiri adalah sebuah peralatan untuk mengambil foto (biasanya terdiri dari sebuah kotak yang tanpa cahaya dengan sebuah lensa pada sebuah sisi dan pada sisi lainnya terdapat film yang peka akan cahaya. Kamera juga dapat berarti perangkat televisi yang terdiri dari sebuah sistem lensa yang memfokuskan gambar ke mosaik yang sangat peka terhadap cahaya yang di *scan* dengan tembakan elektron (http://hyperdictionary.com/dictionary/camera).

2.2 *Image*

Image adalah semua jenis file yang disusun dari piksel (Anonim). *Image* atau citra merupakan representasi dari suatu bentuk. Saat ini pemrosesan *image* menjadi semakin baik karena didukung oleh prosesor yang lebih cepat dengan harga yang lebih murah. Citra dapat disimpan dalam berbagai format. Dua tipe citra yang fundamental adalah vektor dan *bitmap*. Citra vektor disimpan sebagai rangkaian instruksi-instruksi gambar, sedangkan citra *bitmap* disimpan sebagai elemen-elemen gambar mosaik.

Jenis format grafik diantaranya, yaitu:

- JPEG (*Joint Photographic Expert Group*) adalah *format file* terkompresi *lossy* untuk menampung foto dari citra yang kontinu.

- GIF (*Graphics Interchange Format*) merupakan *file* terkompresi *lossless* dimana warna maksimum adalah 256 yang mendukung transparansi dan animasi serta bagus untuk gambar yang memiliki area besar berwarna tunggal.
- BMP adalah *format file* standar untuk *Windows*. *Format file* ini mendukung RGB, *grayscale*, *bitmap*, maupun *index color*. BMP terdiri atas *dot-dot* atau *bit-bit* yang tersusun yang akan terbentuk dalam suatu gambar.
- TIFF (*Tagg Image File Format*) merupakan *file format* yang biasa digunakan pada *desktop publishing* dan juga percetakan.
- PICT merupakan *format file* untuk *Mac* yang mana dapat menampung obyek vektor dan *bitmap*.
- EPS (*Encapsulated PostScript*) merupakan *file format* untuk mentransfer grafik (vektor atau *bitmap*) dalam bahasa *PostScript* antar aplikasi.
- PSD merupakan *format* standar *Photoshop* yang mana mendukung semua mode citra dan tidak terkompresi.
- PNG (*Portable Network Graphics*) merupakan *file format* yang mendukung pemakaian dalam *web* dengan 24 *bit* dan *background transparency*.

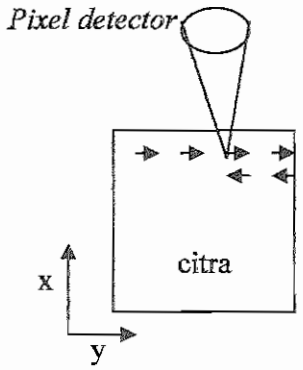
Image processing bertujuan untuk memproses *image*. *Image processing* dapat berupa pengurangan *noise* atau untuk mengambil informasi dari *image* tersebut. Pengambilan informasi tersebut berguna untuk proses selanjutnya.

2.3 Pencarian posisi

A. *Point Scanning*

Dengan menggunakan *detector* piksel (*pixel detector*), sebuah citra dapat diambil mendeteksi satu per satu piksel gambar pada koordinat XY (lihat Gambar 2.1). kelebihan dari sistem seperti ini adalah resolusi yang tinggi, keseragaman ukuran dari satu sisi ke sisi yang lain, dan kesederhanaan

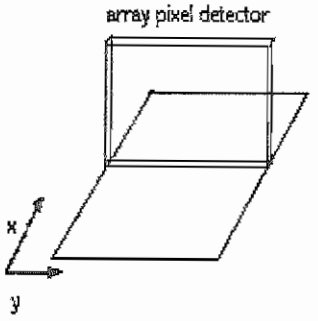
dari *detector* piksel. Kekurangannya kecepatan *scanning* yang lambat (karena harus men-*scan* seluruh piksel.



Gambar 2.1 *Point Scanning*

B. ***Line Scanning***

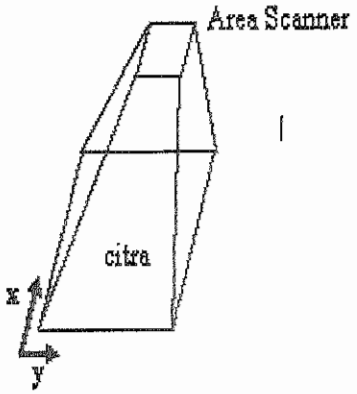
Susunan *detector* piksel yang diletakkan disepanjang satu sumbu (sumbu X atau sumbu Y) sehingga pembacaan hanya 1 arah (pada satu sumbu). Pertama-tama 1 baris informasi dari citra diambil dan dibaca. Setelah itu dilanjutkan pada baris berikutnya. Kelebihan dari sistem ini adalah resolusi yang baik, waktu *scan* yang tidak begitu lama, dan *scanning* pada satu arah. Kekurangannya adalah biaya yang relatif mahal dibandingkan dengan *point scanning*.



Gambar 2.2 *Line Scanning*

C. Area *Scanning*

Detector piksel yang terdiri dari array 2 dimensi yang dibuat sehingga seluruh citra dapat diambil pada waktu yang bersamaan. Kelebihannya waktu *scan* yang cepat. Kelemahannya resolusi yang rendah, dan noise yang tinggi.



Gambar 2.3 Area Scanning

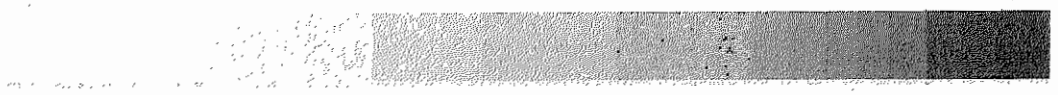
2.4 *Grayscale*

Grayscale adalah derajat keabuan dari suatu citra. Penggunaan citra *grayscale* membutuhkan sedikit informasi yang diberikan pada tiap piksel dibandingkan dengan citra berwarna sehingga lebih memudahkan pemrosesan data dalam *image processing*. Warna abu-abu pada citra *grayscale* merupakan gabungan dari warna merah (*red*), hijau (*green*), dan biru (*blue*) yang memiliki intensitas yang sama.

Dalam citra *grayscale* hanya dibutuhkan intensitas tunggal, yaitu hitam dan putih. Lebih sedikit dibandingkan dengan citra berwarna yang membutuhkan 3 intensitas untuk tiap pikselnya. Derajat keabuan memiliki 256 level atau 8 bit integer. Nilai 0 untuk hitam yang merupakan piksel yang paling lemah intensitasnya dan 255

untuk putih yang merupakan piksel dengan intensitas tertinggi, sedangkan diantaranya adalah nilai untuk derajat keabuan.

Dengan sedikitnya informasi diberikan, pemrosesan citra *grayscale* dapat lebih cepat dibandingkan dengan pemrosesan citra berwarna. Selain itu, memori yang dibutuhkan juga lebih sedikit dibandingkan dengan citra berwarna. Hal ini dikarenakan pada citra *grayscale* hanya terdapat satu nilai intensitas pada tiap pikselnya, sedangkan pada citra berwarna terdapat tiga nilai intensitas (merah, biru, hijau) pada tiap pikselnya.



Gambar 2.4 Gradasi *Greyscale* 256 level

2.5 *Binary Image*

Binary image atau yang juga dikenal dengan sebutan *black and white* merupakan tipe paling dasar dari citra dimana hanya terdiri dari dua warna / nilai saja, yaitu hitam dan putih (1 dan 0). Dimana 1 adalah warna putih dan 0 adalah warna hitam. Terkadang ada yang menyebutnya 0 dan 255 sebagai pengganti 1 (dimana 0 untuk warna hitam dan 1 untuk warna putih).

Binary image sering digunakan dalam pemrosesan citra. Hal ini disebabkan proses *binary image* lebih cepat, dan lebih mudah dibandingkan dengan pemrosesan citra berwarna karena hanya terdiri dari 2 warna saja. Selain itu, dari segi biaya pemrosesan *binary image* jauh lebih murah daripada pemrosesan citra berwarna.

Untuk pemrosesan *binary image* hanya diperlukan 1 *bit* saja, sedangkan untuk *grayscale* diperlukan 8 *bit*. Oleh karena itu memori yang dibutuhkan untuk *binary image* lebih kecil daripada citra *grayscale* dalam pemrosesan citra. Dengan demikian pemrosesan citra *binary image* dapat menjadi lebih banyak kapasitasnya dibandingkan dengan citra *grayscale* apabila menggunakan memori yang besarnya sama.

2.6 *Thresholding*

Thresholding digunakan untuk memisahkan obyek dari latar belakang (*background*). Dengan cara dibuat perbedaan yang mencolok antara *background* dengan obyek. *Thresholding* dilakukan setelah citra di buat *grayscale* terlebih dahulu. Nilai yang lebih besar dari *threshold* adalah obyek dan nilai yang lebih kecil sebagai *background.*

Citra *thresholding* adalah sebagai berikut:

$$T = T(x, y, p(x,y), f(x,y))$$

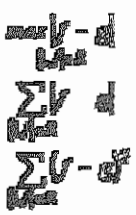
Citra *global thresholding* adalah sebagai berikut:

$$G(x,y) = \begin{cases} 1, \text{ jika } f(x,y) \geq T \\ 0, \text{ lainnya} \end{cases}$$

f(x,y) adalah *grayscale* dari titik (x,y) dan p(x,y) merupakan beberapa properti lokal dari titik ini. Jika T hanya bergantung pada f(x,y) maka disebut *global thresholding*. Jika T bergantung pada f(x,y) dan p(x,y) maka disebut *local thresholding*. Secara umum *thresholding* memberikan hasil yang maksimal bila puncak *histogram* tinggi / sempit dan dipisahkan oleh lembah yang dalam (Fu, Gonzales, dan Lee, 1987, p363).

2.7 *Template Matching*

Template matching adalah metode pemrosesan citra dengan cara membandingkan citra dengan *template* yang ada. Untuk mendeteksi *image* f[i,j] dengan *template* g[i,j] adalah dengan menempatkan *template* tersebut pada lokasi *image* dan mendeteksi titik tersebut dengan nilai intensitas yang terdapat pada *template*. Karena sangat sulit mendapatkan nilai yang sama persis antara *template* dan *image*, maka harus dihitung ketidaksamaan antara nilai intensitas pada *template* dan nilai yang berkaitan pada *image* tersebut.



dimana R adalah *region* dari *template*.

jumlah dari *error* adalah cara yang paling sering digunakan, pada kasus *template matching* ini rumusnya adalah sebagai berikut

$$\sum_{x,y}[f(x,y)-g(x,y)]^2=\sum_{x,y}f(x,y)^2+\sum_{x,y}g(x,y)^2-2\sum_{x,y}f(x,y)g(x,y)$$

Dengan menganggap bahwa f dan g bernilai tetap kemudian $\sum fg$ memberikan ukuran dari ketidaksamaan. Strategi yang digunakan untuk mendapatkan semua lokasi dan *variable* dari *template* tersebut, yaitu dengan cara menggeser *template* tersebut dan menggunakan ukuran yang cocok untuk setiap titik pada *image*. Untuk m x n *template* dapat dilakukan perhitungan sebagai berikut

$$c(k,l)=\sum_{i=1}^{m}\sum_{j=1}^{n}f(i,j)g(i+k,j+l)$$

dimana k dan l adalah nilai persimpangan terhadap *template* dari *image*. Operasi ini disebut *cross correlation* antara f dan g. Tujuannya adalah mendapatkan lokasi yang mendapat nilai diatas nilai *threshold*.

2.8 Filter

Suatu citra yang diambil melalui kamera tidak dapat diproses secara langsung. Hal ini dikarenakan masih terdapat berbagai intensitas, kontras, dan juga *noise* sehingga proses pengenalan obyek menjadi sulit. Untuk menghilangkan *noise* dan untuk mempermudah pemrosesan citra maka dilakukan proses pemfilteran.

Fu (1987, p162) mengatakan filter yang digunakan dalam proses pemfilteran citra adalah *median* filter dan *mean* filter. Dalam hal ini setiap piksel *grayscale* diubah dengan nilai rata-rata (*mean*) dari sekeliling pikselnya, tergantung *mask* yang digunakan apakah 3x3 atau 5x5 dan sebagainya. *Median filter* menggunakan *median* dari *mask* yang digunakan, yaitu nilai tengah dari jumlah data *mask* yang digunakan. Misal *mask* 3x3, maka mediannya adalah data ke 5. Bentuk *mask* 3x3 adalah sebagai berikut:

$$\begin{bmatrix} (x-1, y-1) & (x-1, y) & (x-1, y+1) \\ (x, y-1) & (x, y) & (x, y+1) \\ (x+1, y-1) & (x+1, y) & (x+1, y+1) \end{bmatrix}$$

$$\begin{bmatrix} (1,1) & (1,2) & (1,3) \\ (2,1) & (2,2) & (2,3) \\ (3,1) & (3,2) & (3,3) \end{bmatrix}$$

2.9 *Fast Fourier Transform* (FFT)

Fast Fourier Transform adalah bentuk penyederhanaan dari DFT (*Discrete Fourier Transform*). DFT dan FFT memiliki rumus umum yang sama, yaitu:

$$f_i = \sum_{k=0}^{n-1} x_k e^{\frac{-2\pi}{n} j_k}$$

f = Fourier Transform

x = bilangan kompleks

j = 0,1,2, ... , n-1

i = imajiner

n = banyaknya bilangan

e = bilangan natural

tetapi penghitungannya berbeda. Pada DFT perhitungan dilakukan secara *direct computation* dengan jumlah perhitungan sebanyak N^2, sedangkan pada FFT perhitungan hanya N log N.

Dalam suatu sinyal periodik (berulang), suatu titik antara periode yang satu dengan yang lain memiliki nilai yang sama (faktor *twiddle*). Dengan demikian terdapat nilai-nilai yang sama yang tidak perlu dihitung kembali. Algoritma FFT mencegah suatu nilai yang sama dihitung dua kali.

Kemudian semua nilai yang sama akan dijumlahkan. Hal inilah yang menyebabkan FFT menjadi lebih cepat daripada DFT. Oleh karena itu penghitungan DFT menghemat memori, dan lebih sederhana dibandingkan dengan DFT.

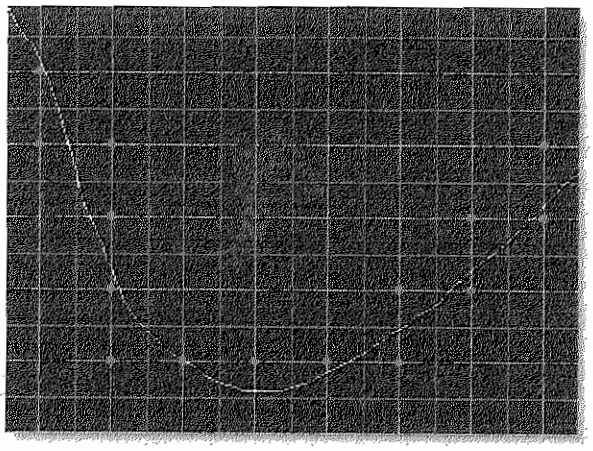
2.10 Freeman *Chain Code*

Gambar garis adalah sebuah citra yang berupa garis atau segmen-segmen kurva yang terhubung atau tidak terhubung. Beberapa aplikasi yang membutuhkan gambar garis, misalnya: komputer grafis, pemrosesan citra, pengenalan pola, dan kartografi otomatis pada sistem informasi geografi dimana kontur dari peta harus dimasukkan.

Freeman memperkenalkan metode yang dinamakan *Chain Code* untuk merepresentasikan gambar garis. Pengkodean ini effisien dalam transmisi dan penyimpanan data. Oleh karena itu dibutuhkan kuantisasi dan *encode* dari citra 2D. Sebuah *mesh* seragam diletakkan pada gambar garis. Diasumsikan jarak antara dua *mesh* yang berurutan adalah T. Sebuah gambar garis dikuantisasi menggunakan titik-titik kurva. Titik-titik kurva yang terpilih untuk merepresentasikan titik-titik dari gambar garis. Terdapat 3 metode kuantisasi , yaitu:

a. Kuantisasi Kotak

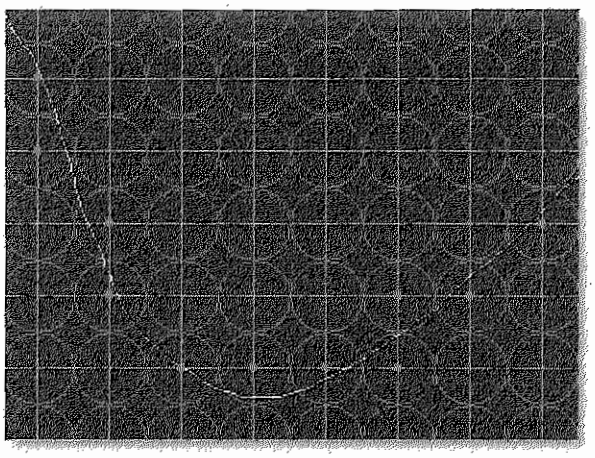
Setiap *node mesh* diambil dan diletakkan ditengah-tengah kotak dari sisi T seperti ditunjukkan pada Gambar 2.5. Ketika gambar garis atau kurva jatuh dalam kotak yang merupakan bagian dari *node mesh*, maka *node* tersebut akan ditandai sebagai titik kurva.



Gambar 2.5 Kuantisasi Kotak

b. Kuantisasi lingkaran

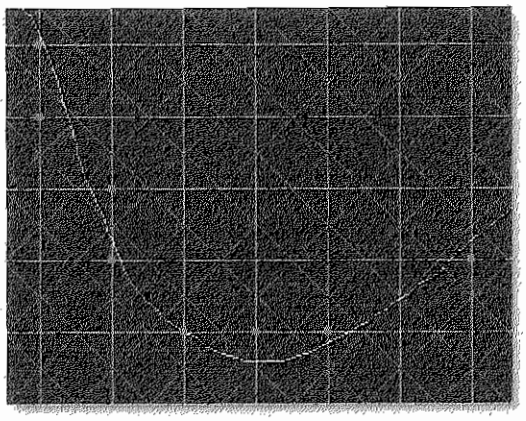
Cara yang sama seperti kuantisasi kotak digunakan pada kuantisasi lingkaran. Perbedaannya adalah pada kuantisasi ini *node mesh* diletakkan pada pusat lingkaran dengan radius T/2 seperti ditunjukkan pada Gambar 2.6. ketika gambar garis jatuh dalam lingkaran, maka *node mesh* yang terdapat pada pusat lingkaran ditandai sebagai titik kurva.



Gambar 2.6 Kuantisasi Lingkaran

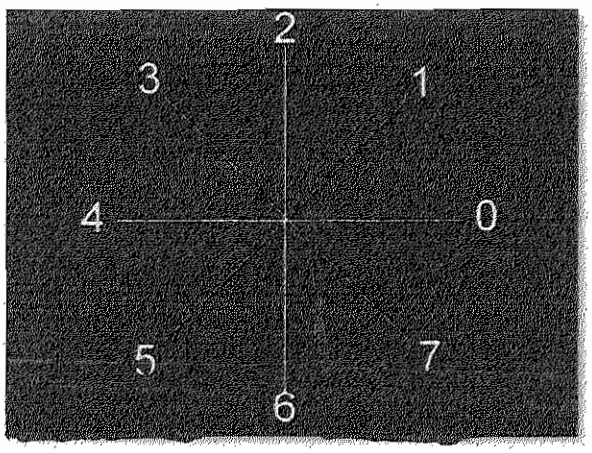
c. Kuantisasi *Grid-Intersect*

Titik-titik kurva dipilih berdasarkan dimana kurva memotong garis *mesh* antara *node-node* yang berdekatan. *Node* yang paling dekat titik potong dipilih sebagai titik kurva. Ketika terdapat lebih dari satu titik potong di sekitar *node mesh*, hanya satu titik kurva yang terpilih.



Gambar 2.7 Kuantisasi *Grid-Intersect*

Pendekatan dari gambar garis menggunakan sekuen dari titik kurva dengan menggunakan cara kuantisasi seperti diatas dapat dikodekan menggunakan 8 bit untuk 1 garis segmen. Ini dikarenakan untuk setiap titik kurva memiliki 8 arah seperti ditunjukkan pada gambar 2.8.



Gambar 2.8 Arah *Node*

Setiap segmen garis dalam pendekatan diberi sebuah kode berdasarkan 8 arah yang ada. *Chain code* merepresentasikan setiap gambar garis. Perbedaan kuantisasi akan menyebabkan perbedaan *chain code* untuk gambar garis yang sama (Absar, Sattar, 2004).

2.11 *Gaussian Blur*

Merupakan *image-blurring* filter menggunakan distribusi normal (Distribusi *Gaussian*). Persamaan dari distribusi Gaussian, yaitu:

$$Gr = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-r^2}{2\sigma^2}}$$

r = radius blur

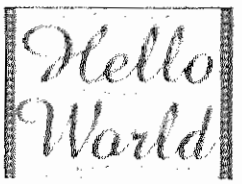
σ = standar deviasi

Gr = distribusi *Gaussian*

Pada distribusi 2D dihasilkan kontur lingkaran dengan pendistribusian nilai dari titik pusat. Piksel-piksel yang mendistribusikan ini tidak nol dan digunakan untuk membuat matriks konvolusi. Setiap nilai piksel mempengaruhi berat rata-rata dari piksel tetangganya. Nilai piksel *original* diterima sebagai yang paling berat (memiliki nilai *Gaussian* yang paling besar), dan piksel tetangganya diterima sebagai yang paling ringan dan berbanding lurus dengan jarak dengan piksel *original*.

Secara teori, distribusi pada setiap titik pada citra tidak akan nol. Ini berarti bahwa seluruh citra harus dimasukkan ke dalam perhitungan untuk setiap piksel. Pada kenyataannya, pada penghitungan dari fungsi *Gaussian*, piksel diluar lebih-kurang 3σ cukup kecil dan bisa dianggap nol. Kemudian piksel-piksel diluar *range* dapat diabaikan. Untuk perhitungan program hanya membutuhkan matriks dengan dimensi [6σ + 1, 6σ + 1] untuk memastikan semua piksel yang relevan terhitung. Keistimewaan dari *Gaussian blur* adalah dapat menerapkan penghitungan matriks satu dimensi kedalam matriks dua dimensi dengan cara menghitung secara horizontal dan mengulang proses tersebut untuk menghitung secara vertical (Ian, 2004).

2.9(a) Citra Original



2.9(b) Citra *Blur*

Fast Gaussian Blur

Metode yang sangat populer digunakan untuk membuat *blur* adalah dengan melakukan konvolusi antara Gaussian *kernel* dengan citra dengan menggunakan *Fast Fourier Transform* (FFT). Implementasi dari FFT biasanya memiliki dua masalah. Masalah yang pertama citra yang tidak negatif ketika di-*blur*-kan dengan FFT menjadi negatif sebagai hasil dari *numerical round-off error*. Masalah yang kedua adalah penggunaan skala yang besar pada *blurring* (David, 2002).

Filter *Gaussian*

Filter *Gaussian* berguna untuk memperhalus (*smooth*) citra sebelum dilakukan deteksi sisi. Filter ini dapat memperhalus sisi. Selain itu juga menghilangkan *noise* dari citra. Rumus untuk filter *Gaussian* adalah

$$F = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij} C_{ij}}{S}$$

F = filter *value* dari piksel target

P = piksel dalam *grid* 2D

C = koefisien filter *Gaussian*

n = dimensi matriks

S = jumlah seluruh nilai dalam matriks Gaussian

0	1	2	1	0
1	4	8	4	1
2	8	16	8	2
1	4	8	4	1
0	1	2	1	0

Gambar 2.10 Contoh Gaussian *Mask* 5x5

(Anonim).

2.12 Canny *Edge Detection*

Pendeteksian sisi merupakan hal penting dalam pemrosesan citra, dimana sisi pada suatu citra merupakan area dengan beda intensitas piksel yang satu dengan piksel tetangganya besar. Pendeteksian sisi pada suatu citra secara signifikan mengurangi jumlah piksel dari citra dan menyaring informasi pada citra yang tidak terpakai. Beberapa kriteria yang harus ada pada Canny *edge detection* yaitu:

1. *Error rate* yang rendah.
2. Titik-titik pada sisi dapat ditempatkan secara baik.
3. Hanya mempunyai satu buah respon untuk satu buah sisi.

Langkah-langkah dalam melakukan Canny *edge detection*, yaitu:

1. Melakukan *smoothing* pada citra dengan menggunakan filter *Gaussian.*
2. Mencari *magnitude* dari gradien-gradien pada citra dan memberikan tanda pada *region* yang mempunyai beda intensitas yang besar.
3. Melakukan penyusuran pada *region* tersebut dan menurunkan nilai piksel yang tidak maksimum tersebut menjadi 0 (*Non-maximum Suppression*).
4. Melakukan pengurangan pada *array* gradien dengan menggunakan histeresis.

1. *Smoothing* dengan filter *Gaussian*

Dalam melakukan Canny *edge detection* ada beberapa langkah yang harus dilakukan. Langkah pertama yang harus dilakukan adalah melakukan *smoothing* pada citra dengan menggunakan filter *Gaussian*, yang berfungsi untuk menghilangkan *noise* pada citra sebelum dilakukan *edge detection*. Penggunaan filter *Gaussian* untuk melakukan *smoothing* dikarenakan filter *Gaussian* ini terdiri dari *mask* sederhana aja.

Smoothing menggunakan filter *Gaussian* ini dapat dilakukan dengan cara melakukan konvolusi yaitu menjalankan *mask* pada citra. *Mask* harus mempunyai ukuran lebih kecil daripada citra, semakin besar *mask gaussian*,

maka semakin tidak sensitif terhadap *noise*. *Gaussian mask* yang digunakan dalam penelitian ini adalah sebagai berikut

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Gambar 2.11 *Gaussian mask* 5x5 dengan $\sigma = 3{,}78$

2. **Mencari *magnitude* dari gradien yang didapat menggunakan operator Sobel**

Setelah melakukan proses *smoothing* dan mengeliminasi *noise* yang ada pada citra, maka langkah selanjutnya adalah mencari gradien pada citra tersebut. Pencarian gradien dilakukan dengan cara melakukan konvolusi pada citra, yaitu dengan menjalankan *mask* dari Sobel dalam 2 arah yaitu arah x dan arah y. Operator Sobel yang digunakan dalam penelitian ini adalah sebagai berikut:

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

Gx

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Gy

Gambar 2.12 (a) Sobel operator arah x, (b) Sobel operator arah y

Magnitude atau *edge strength* dari gradien-gradien yang didapatkan, dapat dihitung dengan rumus:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

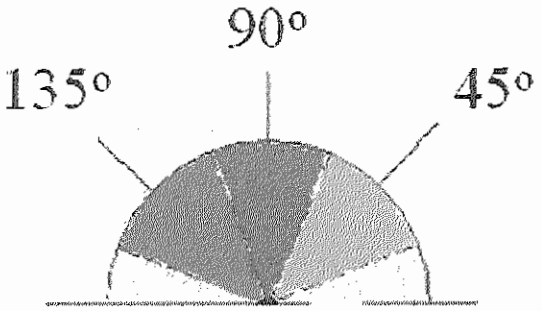
3. **Mencari *edge direction***

Pencarian arah dari suatu *edge* sangat penting setelah gradien dari arah x dan y didapatkan. Pencarian *edge direction* dapat digunakan rumus sebagai berikut:

theta = invtan (Gy / Gx)

4. **Melakukan normalisasi dari *edge direction***

Edge direction yang digunakan dalam penelitian ini ada 4 yaitu 0 derajat, 45 derajat, 90 derajat, 135 derajat.



Gambar 2.13 *Edge direction*

untuk *edge direction* yang jatuh dalam range kuning (0-22.5 dan 157.5-180) akan dilakukan normalisasi menjadi 0 derajat. Untuk *Edge Direction* yang jatuh dalam range hijau (22.5-67.5) akan dilakukan normalisasi menjadi 45 derajat. Untuk *Edge Direction* yang jatuh dalam range biru (67.5-112.5) akan dilakukan normalisasi menjadi 90 derajat dan untuk *Edge Direction* yang jatuh dalam range merah (112.5-157.5) akan dilakukan normalisasi menjadi 135 derajat.

5. **Melakukan proses *non-maximum Suppression***

Setelah proses normalisasi dari *edge direction* selesai, kemudian dilakukan proses *non-maximum suppression*. Proses *non-maximum suppression* adalah melakukan penelusuran terhadap *region-region* gradien yang didapat dari setiap piksel, jika nilainya tidak maksimum akan dilakukan penurunan nilai pikselnya menjadi 0.

6. **Melakukan pengurangan pada *array* gradien dengan menggunakan histeresis**

Proses histeresis digunakan untuk mengeliminasi *streaking* (garis-garis yang tebal), proses ini berguna untuk melakukan *thinning* pada citra, sehingga lebar tepi menjadi 1 piksel saja. Pada proses histeresis ini ada 2 buah nilai *threshold* yaitu T1 (*threshold* nilai rendah) dan T2 (*threshold* nilai tinggi).

Jika nilai piksel < T1 maka nilai piksel akan diset menjadi 0

Jika nilai piksel > T2 maka nilai piksel akan diset menjadi 255

Jika nilai piksel > T1 dan < T2 maka nilai piksel akan diset menjadi 0.

(Green, 2002)

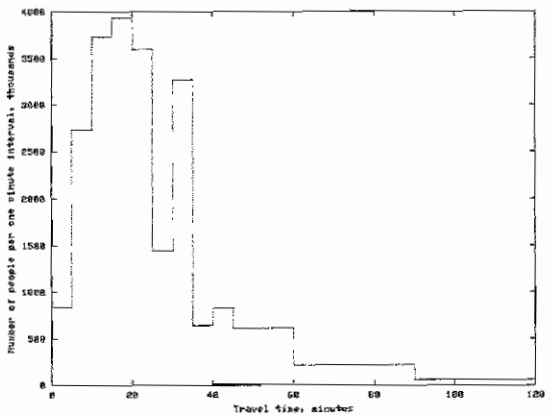
2.13 Histogram

Histogram berasal dari kata Yunani, yaitu histo yang berarti jaringan dan gram yang berarti tulisan atau gambar. Akhiran gram memiliki arti yang sama dengan *graph* yang mengacu kepada menulis atau menggambar. Sekarang ini, histogram lebih dikenal sebagai grafik yang menggambarkan distribusi tingkat warna abu-abu pada sebuah gambar (terutama gambar digital).

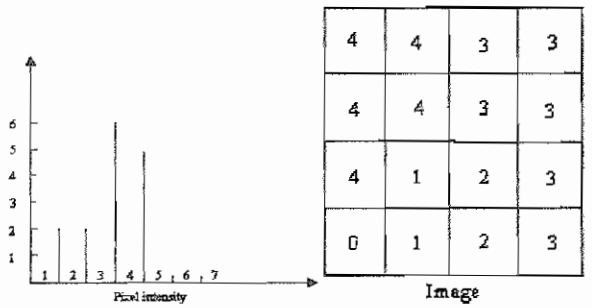
Menurut Aldrich dari *Departement of Economics of the University of Southampton*, orang membuat istilah histogram menjadi terkenal adalah Karl Pearson. Karl menggunakan istilah histogram sebagai bentuk umum dari grafik yang mempresentasikan sebuah nilai dalam kuliah statistiknya.

Histogram dari sebuah gambar adalah grafik yang menggambarkan intensitas yang dimiliki oleh gambar tersebut. Histogram menyediakan informasi mengenai kontras dan distribusi intensitas warna. Biasanya histogram merupakan

grafik dua dimensi, dimana sumbu x adalah tingkat keabuan dan sumbu y adalah intensitasnya. Tingkat keabuan dimulai dari 0 sampai 255, dimana 0 merupakan warna hitam dan 255 merupakan warna putih.



Gambar 2.14 Histogram



Gambar 2.15 Contoh penggunaan histogram untuk distribusi tingkat warna

Gambar diatas merupakan histogram dari suatu citra. Dari histogram tersebut dapat diperoleh informasi jumlah piksel yang memiliki nilai 0 sama dengan 1, jumlah piksel yang memiliki nilai 1 sama dengan 2, jumlah piksel yang memiliki nilai 2 sama dengan 2, jumlah piksel yang memiliki nilai 3 sama dengan 6, dan jumlah piksel yang memiliki nilai 4 sama dengan 5. dari informasi yang diperoleh dapat menentukan batas threshold yang akan digunakan.

2.14 Ragam (*Variance*)

Ragam digunakan untuk mengukur keragaman dari suatu data yang didapat. Ragam memperhatikan posisi relatif setiap pengamatan terhadap nilai tengah dari gugus data yang ada.

Walpole (1995, p33) mengatakan Ragam dengan populasi bernilai terhingga dan berukuran N dapat dilambangkan dengan δ^2 dapat dihitung dengan rumus :

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}$$

σ^2 = ragam (varian)
N = jumlah data
x_i = nilai x yang ke i
μ = rata-rata

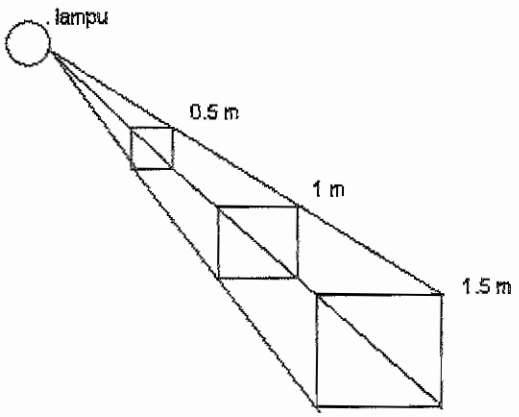
Nilai μ (rata-rata) dapat dicari dengan rumus:

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N}$$

2.15 Pencahayaan

Terang gelapnya suatu gambar (citra) bergantung pada cahaya yang mengenainya. Untuk mengambil gambar yang kualitasnya baik dan tidak terlalu terang tergantung dari jarak pencahayaan. Oleh karena itu untuk mendapatkan gambar dengan kualitas yang baik diperlukan suatu sistem pencahayaan yang baik pula.

Intensitas cahaya diukur dalam satuan *candela* atau *lux*. Jarak pencahayaan ini tergantung dari *power source* dan juga jarak antara lampu dengan obyek. Semakin jauh obyek maka semakin besar *power source* yang dibutuhkan untuk menghasilkan tingkat intensitas yang sama.



Gambar 2.16 Semakin jauh jarak maka semakin besar *power source* untuk mendapatkan tingkat intensitas yang sama

2.16 *Charge Coupled Device* (CCD)

Charge Coupled Device adalah rangkaian integrasi peka cahaya yang menyimpan dan menampilkan data untuk *image* dimana tiap piksel dari *image* di konversi kedalam muatan listrik dimana intensitas dari muatan listrik tersebut berhubungan dengan spectrum warna (Anonim, 2000). *Charge Coupled Device* terdiri dari kumpulan semikonduktor yang bersifat seperti kapasitor yang saling terhubung atau tergandeng. Dibawah kendali rangkaian eksternal, tiap kapasitor dapat mengirimkan muatan listrik ke kapasitor yang ada disebelahnya. *Charge Coupled Device* terdiri dari deretan kapasitor tunggal yang dapat digunakan sebagai *delay line*. Tegangan analog yang diberikan pada kapasitor pertama dalam *array*, dan kemudian secara tetap perintah diberikan kepada tiap kapasitor untuk mengirimkan muatannya ke kapasitor yang ada disebelahnya sehingga semua *array* mengalami pergeseran sebanyak satu lokasi. Setelah penundaan yang terjadi sama dengan jumlah kapasitor dikalikan dengan interval pergeseran, muatan yang merupakan sinyal *input* sampai pada kapasitor terakhir pada *array*, dimana tegangan ini akan dikuatkan untuk menjadi sinyal *output*.

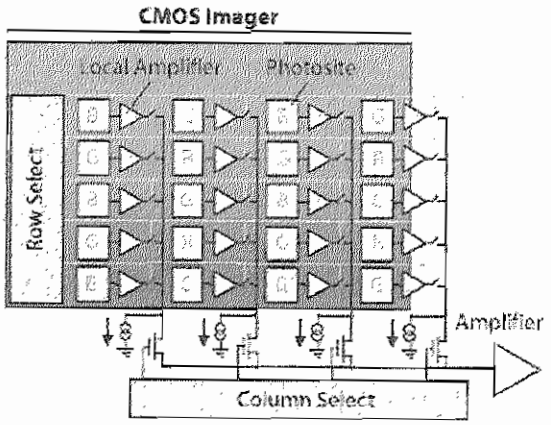
Proses ini berlangsung untuk jangka waktu yang tidak terbatas. Sehingga menimbulkan sinyal pada *output* yang dikarenakan penundaan dari sinyal *input*, serta beberapa distorsi pada saat proses *sampling*. CCD umumnya dapat merespon 70%

cahaya, hal ini membuat CCD lebih efisien dibandingkan dengan *photographic* film yang hanya dapat menangkap kira-kira 2% cahaya.

Gambar diproyeksikan oleh lensa ke *array* kapasitor, menyebabkan tiap kapasitor mengumpulkan muatan yang sebanding dengan intensitas cahaya pada lokasi tersebut. *Array* satu dimensi yang digunakan pada *line scan* kamera, menangkap satu bagian dari gambar sedangkan *array* dua dimensi digunakan pada kamera video, menangkap semua gambar. Pada saat *array* dibuka untuk menerima gambar, rangkaian pengendali menyebabkan tiap kapasitor mengirimkan muatannya kepada kapasitor yang ada disebelahnya. Kapasitor terakhir akan membuang muatannya kepada *amplifier* untuk diubah menjadi tegangan. Dengan mengulang proses ini, rangkaian pengendali akan mengubah semua isi dari *array* menjadi tegangan yang bervariasi, yang akan di *sample*, di digitalisasi, dan disimpan kedalam memori (Anonim, 2002).

2.17 *Complementary Metal Oxide Semiconductor* (CMOS)

Sensor citra CMOS memiliki fungsi yang banyak, seperti: *image processing*, *edge detection*, *noise reduction* dan *Analog to Digital Converter* (ADC) dalam 1 *chip* tunggal. Selain itu sensor CMOS juga mengurangi junlah komponen eksternal yang terhubung kepadanya, untuk aplikasi pada kamera digital sensor CMOS ini tidak memerlukan lagi *chip* tambahan seperti digital *signal processing*, dan ADC.



Gambar 2.17 Arsitektur CMOS

Teknologi sensor CMOS mulai diluncurkan pada awal tahun 1990-an. Pada sensor ini terdapat *Active Pixel Sensor* (APS). APS ini menambahkan *readout amplifier transistor* kepada tiap piksel. Tujuannya adalah untuk merubah muatan menjadi tegangan dan hal ini terjadi pada tiap piksel. Selain itu APS juga dapat digunakan untuk *random access* kepada sensor yang ada pada tiap piksel, tekniknya sama dengan akses memori pada RAM.

Muatan di-*readout* dari sensor AP CMOS dengan menggunakan rangkaian paralel, yang memperbolehkan sinyal dari piksel tunggal atau dari kumpulan piksel dialamatkan secara langsung. Kemampuan *direct access random* ini memperbolehkan CMOS untuk memilih kumpulan muatan piksel mana yang akan di-*readout* terlebih dahulu. Kemampuan ini disebut dengan *windowing readout*. Sebuah sensor CMOS memiliki kemampuan untuk mengurangi ukuran dari citra. Selain itu sensor ini juga menawarkan potensi peningkatan kecepatan *readout*.

Sebagai tambahan untuk penguatan didalam piksel, rangkaian *amplifier* dapat diletakkan sepanjang rangkaian CMOS. *Amplifier* dapat menghasilkan penguatan yang menyeluruh pada rangkaian CMOS sehingga meningkatkan sensitivitas pada situasi pencahayaan yang sedikit (Ziff Davis Inc, 2001).

2.18 Perbandingan CCD Dan CMOS

Sensor gambar CCD dan CMOS merupakan dua teknologi yang berbeda untuk menangkap gambar secara digital. Kedua teknologi ini sering dipandang sebagai saingan. CCD dan CMOS memiliki kekuatan dan kelemahan yang unik, yang membuat keduanya cocok untuk aplikasi yang berbeda.

Kedua teknologi ini mengubah cahaya menjadi muatan listrik dan memprosesnya menjadi sinyal elektronik. Pada sensor CCD, setiap muatan pada piksel dikirim melalui *node output* yang terbatas untuk diubah menjadi tegangan, di *buffer* kemudian dikirim kepada *chip* sebagai sinyal analog. Semua piksel dapat disediakan untuk menangkap cahaya, dan tingkat keseragaman dari *output* tinggi.

Dalam sensor CMOS, tiap piksel memiliki konversi muatan menjadi tegangan masing-masing dan sensor juga memiliki rangkaian digital, sehingga *output* dari *chip* adalah *bit* digital. Fungsi lainnya adalah mengurangi area untuk

penangkapan cahaya, dan karena tiap piksel melakukan konversi sendiri maka tingkat keseragaman dari *output* rendah. Namun CMOS hanya memerlukan rangkaian yang sedikit untuk melakukan operasi dasar (Dalsa Corp., 2004).

Tabel 2.1 Perbandingan fitur CCD dan CMOS

Fitur	CCD	CMOS
Sinyal keluar dari piksel	Paket elektron	tegangan
Sinyal keluar dari *chip*	Tegangan (analog)	*Bit* (digital)
Sinyal keluar dari kamera	*Bit* (digital)	*Bit* (digital)
Faktor pengisi	Tinggi	Menengah
Perbandingan penguatan	-	Menengah
Sistem *noise*	Rendah	Tinggi
Kompleksitas sistem	Tinggi	rendah
Kompleksitas sensor	Rendah	Tinggi
Komponen kamera	PCB, lensa, kumpulan *chip*	*Chip*, lensa
Biaya	Tergantung pada aplikasi	Tergantung pada aplikasi

Tabel 2.2 Perbandingan CCD dan CMOS dari segi unjuk kerja

Unjuk Kerja	CCD	CMOS
Tingkat respon	Menengah	Sudah baik
Jarak dinamik	Tinggi	Menengah
Keseragaman	Tinggi	Rendah
Keseragaman pengaturan cahaya	Cepat	Lambat
Kecepatan	Sedang menuju cepat	Sangat cepat
Windowing	Terbatas	Luas
Antiblooming	Tinggi	Tinggi
Biasing dan *clocking*	Banyak, tegangan tinggi	Tunggal, tegangan rendah

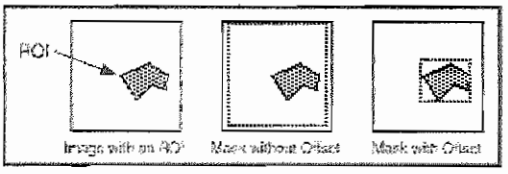
2.19 *Region of Interest* (ROI)

Region of Interest adalah suatu bagian dari citra yang dipilih untuk kemudian diproses. Daerah tersebut dibedakan dengan menggunakan klasifikasi dan *masking*. Jika piksel pada *mask* tidak nol, maka pemrosesan citra dilakukan. Sebaliknya jika piksel pada *mask* sama dengan nol, proses tidak dijalankan.

Setelah daerah yang diinginkan ditemukan, daerah tersebut ditandai dengan kotak untuk membatasi daerah yang akan dikenali. Proses *Region of Interest* (ROI) berbeda dengan *block processing* yang mana memilih bagian citra yang akan diambil untuk diproses. Bagian dari citra dalam *block processing* merupakan bagian citra utuh yang digunakan dalam pengolahan citra.

Dalam *Region of Interest*, citra dapat didefinisikan lebih dari satu *region* (bagian). Bagian tersebut dapat berbentuk poligon yang berupa piksel yang *contiguous* atau berupa *range* dari intensitas. Dengan kata lain, piksel tidak harus selalu *contiguous*.

Region of Interest sangat membantu untuk segmentasi dalam pemrosesan citra karena dengan menggunakan teknik ini citra atau obyek dapat lebih mudah dikenali. Karena obyek sudah akan dibagi dalam *region-region* tertentu sesuai dengan citra obyeknya (Anonim).



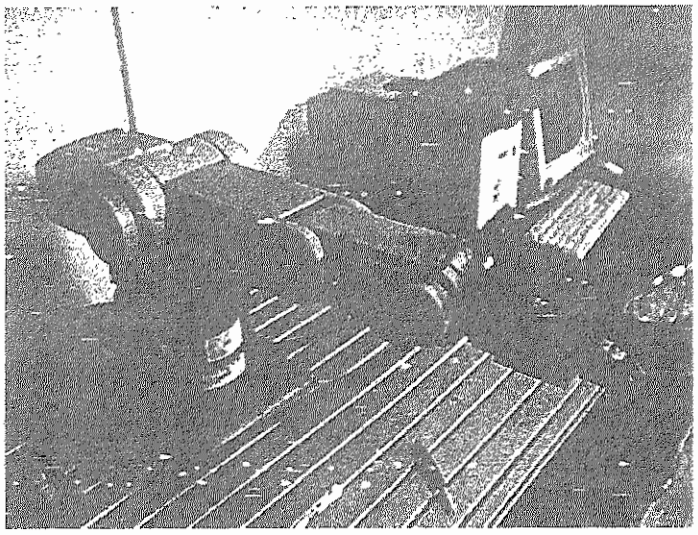
Gambar 2.18 *Region of Interest*

2.20 *Inverse Kinematics*

Kinematika

Kinematika adalah ilmu tentang gerak tanpa memperhatikan penyebabnya. Dari kinematika sehubungan dengan robotika, yang pertama perlu mendapat perhatian adalah matriks rotasi dan matriks transformasi. Matriks rotasi adalah matriks yang memetakan sebuah vektor atau posisi pada suatu sistem koordinat ke sistem koordinat yang lain dalam gerakan rotasi. Matriks transformasi adalah matriks

yang memetakan sebuah vektor atau posisi pada satu sistem koordinat ke sistem koordinat yang lain dengan memperhatikan rotasi, translasi, penskalaan dan perspektif / sudut pandang.



Gambar 2.19 Foto robot RV-M1

Matriks Transformasi *Homogenous*

Matriks transformasi *homogenous* adalah sebuah matriks 4x4, yang mana matriks ini dapat memetakan sebuah vektor posisi dalam koordinat *homogenous* dari suatu sistem koordinat ke sistem koordinat lainnya. Sebuah matriks transformasi *homogenous* terdiri dari 4 submatriks:

$$T = \left[\begin{array}{c|c} R_{3x3} & P_{3x1} \\ \hline f_{1x3} & 1 \text{ X } 1 \end{array}\right] = \left[\begin{array}{c|c} \text{Rotation Matrix} & \text{Position Vector} \\ \hline \text{Perspective Transform} & \text{Scaling} \end{array}\right]$$

Submatriks 3x3 terletak di kiri atas yang merupakan representasi dari matriks rotasi, submatriks 3x1 di bagian kanan atas merepresentasikan vektor posisi. Submatriks 1x3 di bagian kiri bawah merepresentasikan perspektif, dan submatriks 1x1 yang terletak di bagian kanan bawah adalah matriks yang merepresentasikan faktor penskalaan.

Selanjutnya dalam matriks rotasi terdapat 3 macam perputaran, yaitu: perputaran terhadap sumbu x, perputaran terhadap sumbu y, dan perputaran terhadap sumbu z.

θ : sudut pada joint dari sumbu x_{i-1} ke sumbu x_i dengan sumbu z sebagai porosnya (perputaran menggunakan aturan tangan kanan..

d : jarak link ke i - 1 dengan link ke i.

a : jarak / panjang suatu joint dari lengan robot.

α : sudut dari sumbu z_{i-1} ke sumbu z_i dengan poros sumbu x_i (perputaran menggunakan aturan tangan kanan).

Φ : sudut dari sumbu y_{i-1} ke sumbu y_i dengan poros sumbu x_i (perputaran menggunakan aturan tangan kanan).

dx : posisi *end-effector* terhadap sumbu x.

dy : posisi *end-effector* terhadap sumbu y.

dz : posisi *end-effector* terhadap sumbu z.

$$T_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{y,\Phi} = \begin{bmatrix} \cos\Phi & 0 & \sin\Phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Phi & 0 & \cos\Phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

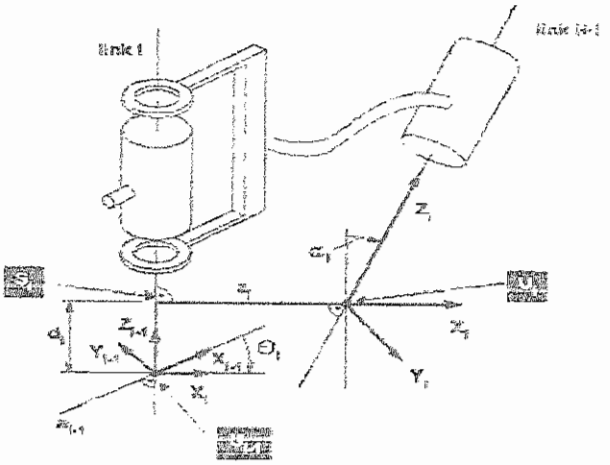
Untuk matriks posisi (translasi):

$$T_{tran} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriks transformasi untuk robot 5 joint adalah

$$^{i-1}T_i = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{i-1}T_i = \begin{bmatrix} \cos\theta & -\cos\alpha\sin\theta & \sin\alpha\sin\theta & a\cos\theta \\ \sin\theta & \cos\alpha\cos\theta & -\cos\theta\sin\alpha & a\sin\theta \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



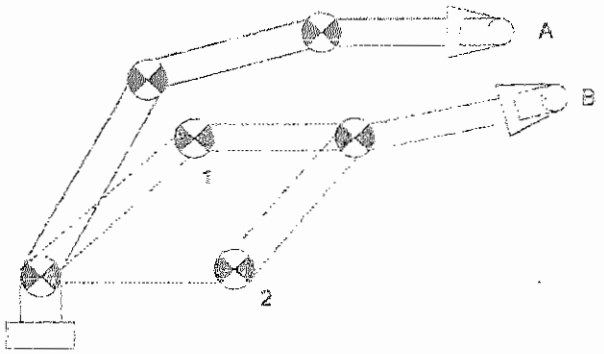
Gambar 2.20 Parameter kinematik menurut konsep Denavit-Hartenberg

Inverse Kinematics

Pada *Inverse Kinematics* pencarian posisi berdasarkan koordinat *end-effector* terhadap koordinat obyek. Setelah koordinat *end-effector* dan obyek diketahui. Lalu robot akan bergerak menuju obyek yang dituju. *Inverse kinematics* digunakan untuk mencari besarnya sudut (θi) yang harus diberikan pada setiap *joint*

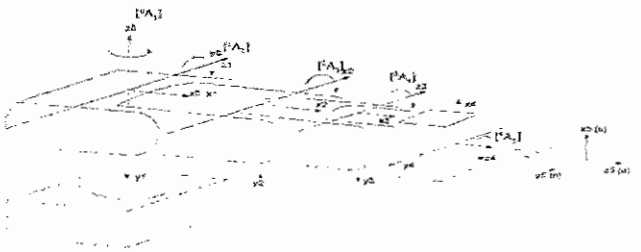
(i) *manipulator* untuk mendapatkan posisi dan orientasi (Fu, Gonzales, Lee, 1987, p52).

Berdasarkan Craig (1989, p114-118) kemungkinan adanya solusi perlu diketahui dahulu sebelum melakukan pencarian nilai θi. Kemudian untuk mendapatkan solusi sangat penting untuk diketahui, hingga perhitungan untuk mencari solusi tidak perlu dilakukan apabila tidak ada jaminan bisa mendapatkan solusi. Ada tidaknya solusi berhubungan dengan area jangkauan (*workspace*) robot. Area jangkauan adalah volum ruang yang dapat dicapai oleh *end-effector manipulator*. Apabila posisi dan orientasi (titik tujuan) dari *end-effector* berada di dalam area jangkauan, maka sekurang-kurangnya terdapat satu solusi. Apabila solusi ada, maka kemungkinan lain yang bisa terjadi adalah solusinya lebih dari satu. Berikut ini adalah contoh posisi *end-effector* dengan solusi lebih dari satu:



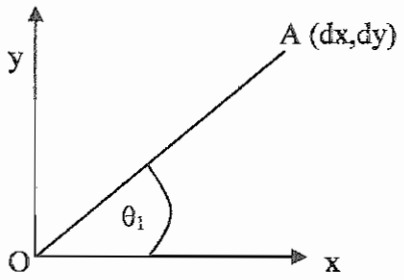
Gambar 2.21 Solusi lebih dari satu

Pendekatan Geometri



Gambar 2.22 Kerangka koordinat robot RV-M1

Metode *inverse kinematics* yang digunakan pada penelitian ini adalah pendekatan geometri. *Sudut* tiap *joint* didapatkan dengan cara melakukan pembandingan antara koordinat obyek dengan *joint* dari lengan robot. Pada pendekatan geometri solusi dicari dengan menerapkan ilmu-ilmu geometri dan hukum-hukum trigonometri. Solusi untuk *joint* 1 adalah sebagai berikut:



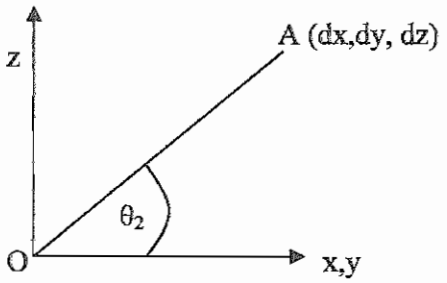
Gambar 2.23 Solusi untuk *joint* 1

Untuk mencari θ_1 digunakan perbandingan antara koordinat x dan y, sehingga didapatkan:

$$\tan\theta_1 = \frac{dy}{dx}$$

$$\theta_1 = \arctan\frac{dy}{dx}$$

Solusi untuk *joint* 2 adalah sebagai berikut:



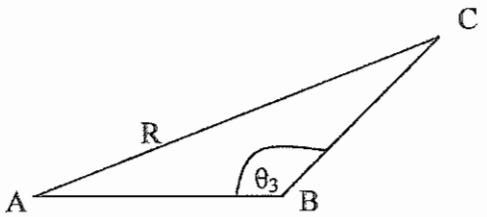
Gambar 2.24 Solusi untuk *joint* 2

Untuk mencari θ_2 digunakan perbandingan antara koordinat x , y dan z, sehingga didapatkan:

$$\tan\theta_2 = \frac{dz}{\sqrt{(dx^2 + dy^2)}}$$

$$\theta_2 = \arctan\frac{dz}{\sqrt{(dx^2 + dy^2)}}$$

Solusi untuk *joint* 3 adalah sebagai berikut:



Gambar 2.25 Solusi untuk *joint* 3

Untuk mencari θ_3 digunakan perbandingan antara koordinat x , y, dan z, serta panjang *joint* dari lengan robot sehingga didapatkan:

$$AC = R = \sqrt{dx^2 + dy^2 + dz^2}$$

$$AB^2 = L_2^2$$

$$BC^2 = L_3^2 + L_4^2$$

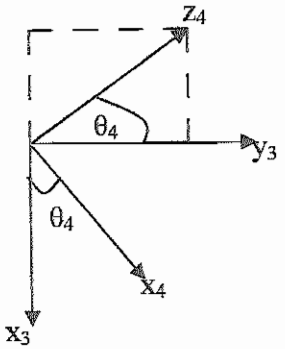
$$\cos\theta_3 = \frac{AB^2 + BC^2 - R^2}{2 \times AB \times BC}$$

$$\cos\theta_3 = \frac{L_2^2 + L_3^2 + L_4^2 - (dx^2 + dy^2 + dz^2)}{2 \times L_2 \times \sqrt{L_3^2 + L_4^2}}$$

$$\theta_3 = \arccos\frac{L_2^2 + L_3^2 + L_4^2 - (dx^2 + dy^2 + dz^2)}{2 \times L_2 \times \sqrt{L_3^2 + L_4^2}}$$

Ln = panjang *link* ke n

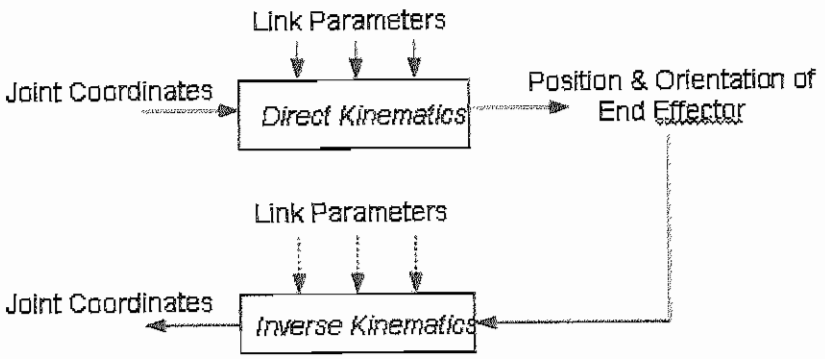
Solusi untuk *joint* 4 adalah sebagai berikut:



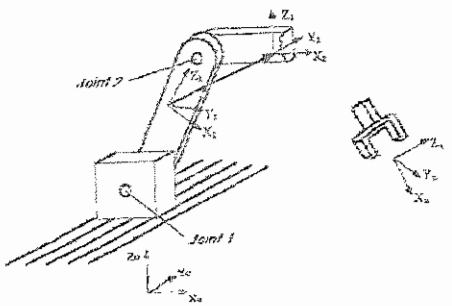
Gambar 2.26 Solusi untuk *joint* 4

$$\tan\theta_4 = \frac{\sin^2\theta_1 - \cos^2\theta_1}{-2\times\cos\theta_1\times\sin\theta_1\times\cos(\theta_2+\theta_3)}$$

$$\theta_4 = \arctan\frac{\sin^2\theta_1 - \cos^2\theta_1}{-2\times\cos\theta_1\times\sin\theta_1\times\cos(\theta_2+\theta_3)}$$



Gambar 2.27 *Direct Kinematics* dan *Inverse Kinematics*



Gambar 2.28 *Links of a Kinematics chain*