BAB II

LANDASAN TEORI

2.1 Visual Basic.Net

Visual Basic.Net 2008 (VBNet 2008) adalah salah satu program berorientasi objek yang diproduksi oleh Microsoft Corp. Program ini biasanya dipaketkan bersama-sama dengan *Visual* C# 2008 dan *Visual* C++ 2008 dalam paket *Visual Studio* 2008. Teknik pemrograman pada VBNet 2008 dapat dibuat lebih terstruktur dibandingkan dengan versinya yang terdahulu, yaitu *Visual Basic* 6 (VB 6) (Wardana, 2008:11).

2.1.1 *Variable*

Variable berguna sebagai penyimpan nilai sementara untuk dapat dipergunakan kembali. Dikatakan sementara karena nilai *variable* akan disimpan dalam memori komputer yang bersifat tidak permanen. Menurut Leong M, (2004:51) *Visual Basic Net* memiliki beberapa aturan dalam memberikan nama pada sebuah *variable*, diantaranya :

1. Nama *variable* harus diawali dengan karakter huruf, dan penamaannya tidak boleh menggunakan tambahan karakter khusus lain kecuali dengan menggunakan *underscore* (_).
2. Panjang maksimal karakter adalah 255.
3. Sifatnya unik sehingga tidak boleh ada dua deklarasi *variable* yang sama dalam prosedur.
4. Tidak mengandung perintah eksekusi yang dikenal oleh VBNet.

VBNet menyediakan pustaka yang berisi fungsi konversi untuk mempermudah melakukan konversi *variable* dengan tipe data tertentu ke tipe data lain. Beberapa fungsi dalam pustaka VBNet tersebut adalah:

Tabel 2.1. Fungsi dalam Pustaka VBNet

Fungsi	**Hasil Konversi**
Cbool	*Boolean*
Cbyte	*Byte*
Cchar	*Char*
Cdate	*Date*
CDbl	*Double*
Cint	*Integer*
CLng	*Long*
CObj	*Object*

2.1.2 Tipe Data

Tipe data adalah jenis nilai yang tersimpan dalam *variable*, bisa berupa huruf, angka, ataupun tanggal. Tipe data diperlukan agar VBNet dapat mengenali jenis data yang tersimpan dalam *variable*. Menurut Leong M, (2004:48) berikut beberapa jenis tipe data dan jangkauan nilai yang didukung oleh VBNet:

Tabel 2.2. Nilai dan Tipe Data

Tipe Data	**Range**
Boolean	Hanya dapat diisi dengan TRUE (benar) atau FALSE (salah)
Byte	0 – 255
Char	0 – 65535
Date	Merupakan nilai sebuah tanggal dan waktu 1 januari 0001 – 31 desember 9999, contoh: *Dim Tgl as Date* *Tgl = #9/16/2008 19:20:20#*
Decimal	0 - +/- 79.228.162.514.264.337.593.543.950. 335 (tanpa bilanagn decimal di blakang koma) atau 0 - +/- 7,9228162514.264337593543950 335 (dengan bilangan decimal di blakang koma), contoh: *Dim Nilai as Decimal* *Nilai = 100,5*
Double	-1,79769313486231570E+308 - 1,7976931348 6231570E+308.

Tipe Data	Range
Integer	-2.147.483.648 - 2.147.483.648.
Long	-9.223.372.036.854.775.808 - 9.223.372.036.854.775.807
Sbyte	-128 – 127.
Short	-32.768 - 32.767.
Single	-3,4028235E+38 - -1,401298E-45 (untuk bilangan negatif) 1,401298E-45 - 3,4028235E+38 (untuk bilangan positif).
String	0 – 2 milyar karakter.
Object	Tipe data umum (sama seperti varian) yang dapat menampung berbagai tipe data lainnya.

2.1.3 Operator Matematika

Menurut Leong M (2004:163) VBNet telah merangkum fungsi matematika secara lengkap dalam *class* khusus yaitu *class Math*. Di dalam *class Math* dapat ditemukan banyak fungsi matematika yang berguna, misalnya fungsi trigonometri, logaritma, dan lain-lain. Beberapa fungsi pada *class Math* dapat dilihat pada Tabel 2.3.

Tabel 2.3. Fungsi pada *Class Math*

E	Bilangan natural atau e = 2,7182818284590452354
PI	Konstanta diameter lingkaran yaitu phi = 3,14159265358979323846
Abs	Fungsi absolut dari bilangan
Acoc	Fungsi sudut dari cosinus bilangan
Asin	Fungsi sudut dari sinus bilangan
Atan	Fungsi sudut dari tangent bilangan
Atan2	Fungsi sudut dari tangen yang ditetapkan dari 2 bilanagn spesifik
Ceiling	Fungsi mencari bilangan terkecil dari angka yang lebih besar atau sama dengan angka yang ditentukan
Cos	Fungsi cosines
Cosh	Fungsi cosinus hiperbola dari suatu sudut
Exp	Fungsi eksponensial
Floor	Fungsi mencari bilangan terbesar dari angka yang lebih besar atau sama dengan angka yang ditentukan
Log	Fungsi log
Log10	Fungsi log10
Max	Mencari nilai maksimum atau terbesar
Min	Mencari nilai minimum atau terkecil
Round	Fungsi pembulatan

Sign	Mencari tanda dari bilangan
Sin	Fungsi sinus
Sinh	Fungsi sinus hiperbola dari sudut
Sqrt	Akar kuadrat dari bilangan
Tan	Fungsi tangent
Tanh	Fungsi tangent hiperbola dari sudut

2.1.4 Koordinat Form

Seluruh *form* pada VBNet memiliki sebuah koordinat yang berguna untuk menentukan posisi gambar atau *graphic* pada *form*. Koordinat tersebut terdiri atas dua bagian yaitu koordinat horizontal dan koordinat vertikal yang biasa dilambangkan sebagai x dan y. Satuan yang digunakan dalam korrdinat *form* adalah *pixel*. Koordinat dimulai dari bagian kiri atas *form* yang merupakan koordinat dasar dari *form* yang memiliki nilai 0 pada koordinat x dan nilai 0 pada koordinat y (Priyanto, 2009:226).

2.2 Grafis

Grafis pada VBNet membutuhkan sistem operasi GDI+ (*Graphic Device Interface*) yang digunakan digunakan sebagai media yang dapat menjadi kanvas untuk menampilkan atau membuat gambar.

2.2.1 GDI+

GDI+ (*Graphic Device Interface*) adalah suatu *class* yang berfungsi untuk memproses sebuah grafis pada VBNet. GDI+ mampu mengolah grafis vektor dua dimensi (garis), *imaging* (*bitmap*) dan *typography* (*font*) pada layar maupun *printer*. Menurut Hendra, dengan adanya GDI+ *programmer* tidak perlu

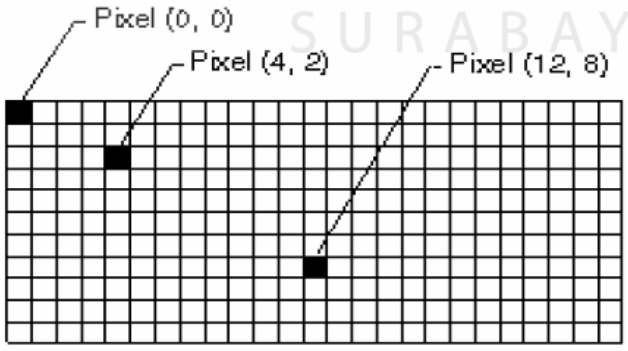
mengetahui detail masing-masing peralatan untuk menampilkan grafis diatasnya, tetapi cukup menggunakan fasilitas yang telah disediakan oleh GDI+ *class*.

2.2.2 Koordinat Tampilan Layar

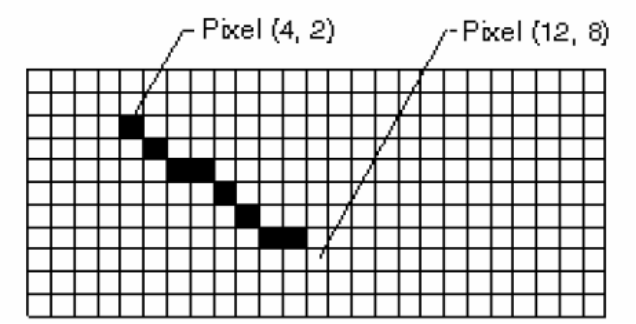
GDI+ menggunakan sistem koordinat, sehingga koordinat tersebut akan ditampilkan di layar (x, y), mulai dari titik koordinat 0, 0.

Gambar 2.1. Koordinat x,y (0, 0) (Hendra, 2011).

Tampilan pada layar komputer koordinat tersebut terbentuk dari *pixel* yang terdapat pada layar komputer.



Gambar 2.2. *Pixel* pada Tampilan Layar Komputer



Gambar 2.3. Proses Pembuatan Garis Koordinat 4,2 sampai 12,8

2.2.3 Gambar Grafis Vektor 2 Dimensi

Grafis vektor 2D merupakan proses penggambaran bentuk-bentuk seperti garis, kurva dan bentuk berdasarkan sekumpulan titik tertentu (*pixel*). Penggambaran objek 2D pada VBNet dapat menggunakan *Object Pen* dimana *pen* dibuat untuk proses pembentukan garis.

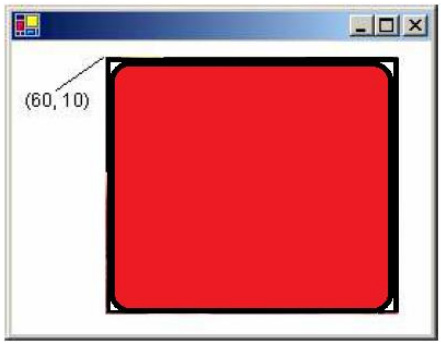
2.3 Penampilan Gambar

Proses penampilan gambar pada VBNet dari file gambar ke *layer*, dapat menggunakan objek yang berada pada VBNet yang berupa *bitmap* dan objek *graphics*. Objek *bitmap* (*class bitmap*) hanya mendukung beberapa format seperti BMP, GIF, JPEG, PNG, dan TIFF.

Contoh pemrograman :

Dim bitmap As New Bitmap (“Grapes.jpg”)

E Graphics DrawImage (bitmap, 60, 10)



Gambar 2.4. Tampilan Contoh Program (Hendra, 2011)

2.4 Posisi Awal Robot

Posisi awal atau proses memperkirakan posisi robot, pada dasarnya menggunakan perhitungan dengan mengakumulasikan data jarak tempuh robot, yang dihasilkan dari sensor robot berupa *rotary encoder* di mana pulsa dari sensor tersebut akan digunakan untuk memperhitungkan jarak tempuh robot. Misalnya dengan menggunakan satuan milimeter. Untuk mendapatkan jumlah pulsa setiap satu kali putaran roda, digunakan rumus sebagai berikut (Ardilla, 2011):

$$Kroda = 2 \times \pi \times r \quad (2.1)$$

$$pulsa\_per\_mm = resolusi\ encoder\ /\ K\ roda \quad (2.2)$$

Pada sistem penggerak *differential* terdapat dua roda yaitu roda kanan dan roda kiri dan dimisalkan jumlah *pulsa_per_mm* untuk roda kanan adalah *right_encoder* dan roda kiri adalah *left_encoder* dan jarak antara dua roda adalah *wheel_base*, maka didapatkan jarak tempuh (*distance*) dan sudut orientasi (θ). Rumusnya adalah sebagai berikut.

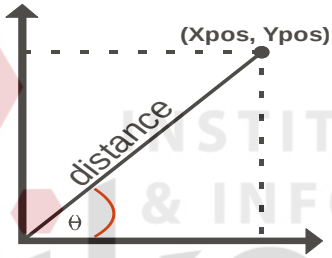
$$distance = (left\_encoder + right\_encoder)/2 \quad (2.3)$$

$$\theta = (left\_encoder - right\_encoder)/wheel\_base \quad (2.4)$$

Karena θ adalah sudut dalam radian maka untuk mengetahui sudut dalam derajat (*heading*) digunakan rumus sebagai berikut :

$$heading = \theta \times \frac{180}{\pi} \quad \text{.....(2.5).}$$

Dari ketentuan di atas didapatkan bahwa nilai *heading* akan bernilai negatif (-) ketika robot berputar melawan arah jarum jam dan akan bernilai positif (+) ketika robot berputar searah dengan jarum jam. Dengan mengetahui jarak dan sudut (*distance* dan θ) maka kita dapat mengetahui koordinat x dan koordinat y dengan persamaan trigonometri sebagai berikut :



Gambar 2.5. Panjang Jarak Koordinat.

Dari ilustrasi pada Gambar 2.5, koordinat dari robot dapat kita ketahui dengan rumus:

$$Xpos = distance \times \sin(\theta) \quad \text{.....(2.6)}$$

$$Ypos = distance \times \cos(\theta) \quad \text{.....(2.7)}$$

2.5 Kesalahan Arah

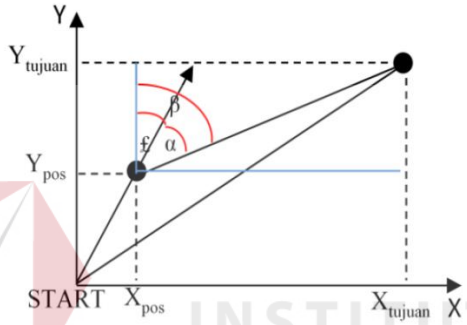
Dalam menentukan *error* arah hadap dari robot terhadap titik tujuan maka digunakan *theorema phytagoras* yang akan menghasilkan posisi (*path*) saat ini dan jarak terhadap titik tujuan (Ardilla, 2011). berikut perhitungannya:

X = X_tujuan – X_pos .....(2.8)

Y = Y_tujuan – Y_pos .....(2.9)

Target_distance = $\sqrt{x^2 + y^2}$ .....(2.10)

Dari *heading* robot yang telah diketahui kita dapat menghitung *error* arah hadap (*heading error*) robot terhadap titik tujuan.



Gambar 2.6. Sudut *α, β,* dan £ (Ardilla, 2011).

Gambar 2.6 menunjukkan ilustrasi untuk mencari *heading error* (α) di mana β adalah *target bearing* yaitu sudut antara posisi robot saat ini terhadap titik tujuan. Sedangkan garis berwarna biru adalah garis bantu yang masing-masing sejajar dengan sumbu x dan sumbu y. Untuk mendapat nilai β, digunakan rumus sebagai berikut :

$$\beta = \arctan\frac{(Y_{tujuan} - Y_{pos})}{(X_{tujuan} - X_{pos})} \quad .....(2.11)$$

Sehingga

$$\alpha = \beta - £ \quad .....(2.12)$$

2.6 *Obstacle Avoidance*

Bagian ini menjelaskan tentang metode yang digunakan untuk menghindari halangan dengan cukup relevan yaitu menggunakan deteksi ujung tepi *object*, tahap kepastian pergerakan terhadap *obstacle*, metode yang *support* terhadap relevansi *obstacle avoidance*.

2.6.1 Metode Analisa Ujung Tepi *Object*

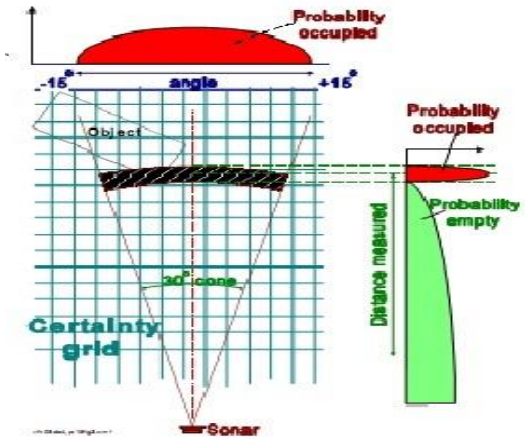
Metode *obstacle avoidance* yang lebih umum dan sering digunakan untuk menghindari rintangan adalah deteksi ujung tepi. Dalam metode ini, algoritma dicoba untuk menentukan posisi dari sisi kendala vertikal dan kemudian mengarahkan robot pada salah satu sisi ujung tepi *obstacle*. Metode ini digunakan dalam penelitian, serta beberapa proyek lainnya semua menggunakan sensor jarak meliputi sensor sharp dan sensor *ultrasonic* untuk mendeteksi *obstacle*. Kelemahan dalam menghindari rintangan yang berbasis pendeteksian ujung tepi adalah robot akan berhenti terlebih dahulu di depan halangan karena proses pengeksekusian perintah terlalu lama. Untuk menanggulangi kelemahan dari pendeteksian tersebut sebaiknya komputer yang digunakan lebih cepat dalam pengeksekusian program agar robot tidak berhenti terlebih dahulu ketika pengambilan data.

Kelemahan lebih lanjut dari metode deteksi ujung tepi adalah sensitivitas untuk sensor akurasi seperti sensor *ultrasonic*, yang sebagian besar digunakan dalam aplikasi *mobile robot*. Dimana posisi minimal untuk batas benda dengan sensor pada posisi spasial adalah 10-50 cm, tergantung pada jarak kendala dan sudut antara permukaan benda.

2.6.2 Kepastian Gerak Terhadap *Obstacle*

Sebuah metode untuk merepresentasikan probabilitas halangan dalam jenis model jaringan telah dikembangkan di Carnegie-Mellon University (CMU). Model metode ini sering disebut dengan kepastian jaringan (*certainty grid*). Secara khusus metode ini sangat cocok untuk akomodasi data sensor yang kurang akurat seperti jarak pengukuran dari sensor *ultrasonic*.

Dalam metode jaringan, area kerja robot diwakili oleh *array* dua dimensi, yang dilambangkan sebagai sel. Setiap sel berisi nilai yang pasti (CV) nilai tersebut menunjukkan bahwa halangan berada dalam wilayah sel tersebut. Dalam perkembangannya metode ini dikembangkan oleh CMU, dengan memperbarui nilai kepastian CV dengan fungsi probabilitas yang memperhitungkan karakteristik sensor yang diberikan. Sebagai contoh sensor *ultrasonic*, yang memiliki pandangan seperti bidang yang berbentuk kerucut. Sebuah sensor *ultrasonic* hanya akan memantulkan gelombang untuk mendapatkan jarak ke objek dalam bidang baca berbentuk kerucut, dan sensor tidak dapat menentukan lokasi sudut objek. (Gambar menunjukkan area A di mana sebuah *obstacle* harus berada dalam *range* untuk menghasilkan pengukuran jarak d).



Gambar 2.7. Sensor Mengenali Benda.

Jika benda terdeteksi oleh sensor *ultrasonic*, hal ini dimungkinkan bahwa objek ini lebih dekat dengan sumbu akustik (gelombang) sensor, daripada bidang kerucut pembacaan sensor *ultrasonic*. Dengan alasan tersebut CMU membuat nilai fungsi Cx dengan meningkatkan nilai CV pada sel yang terdekat pada sumbu akustik lebih dari nilia CV pada sel-sel yang terdapat di pinggir bidang kerucut (pembacaan sensor).

Dalam aplikasi metode CMU ini, mobile robot akan diam saat pembacaan panorama lingkungannya dengan 24 sensor *ultrasonic*. Selanjutnya, fungsi Cx diterapkan pada masing-masing sensor *ultrasonic* dalam upayanya memperbarui informasi jaringan pergerakan. Sehingga, robot dapat bergerak menuju lokasi baru, dan robot akan berhenti kembali untuk mengulangi prosedur tersebut. Setelah robot melintasi ruangan dengan cara ini, kepastian jaringan yang dihasilkan merupakan jalur atau peta yang cukup akurat dari ruangan.

2.7 *Fuzzy Logic Controler*

Logika *fuzzy* adalah cabang dari sistem kecerdasan buatan (*Artificial Intelegent*) yang memanipulasi kemampuan manusia dalam berpikir ke dalam bentuk algoritma yang kemudian dijalankan oleh mesin. Logika *fuzzy* merupakan cara yang tepat untuk memetakan ruang *input* ke dalam ruang *output*. Sebagai contoh:

a. Manajer pergudangan mengatakan pada manajer produksi, seberapa banyak persediaan barang pada akhir minggu ini, kemudian manajer produksi akan menetapkan jumlah barang yang harus diproduksi esok hari.

b. Pelayan restoran memberikan pelayanan terhadap tamu, kemudian tamu akan memberikan tip yang sesuai atas baik tidaknya pelayan yang diberikan.

c. Anda mengatakan pada saya seberapa sejuk ruangan yang anda inginkan, saya akan mengatur putaran kipas yang ada pada ruangan ini.

d. Penumpang taksi berkata pada sopir taksi seberapa cepat laju kendaraan yang diinginkan, sopir taksi akan mengatur pijakan gas taksinya.

Contoh pemetaan *input-output* dalam bentuk grafis terlihat pada Gambar 2.8 berikut.

Gambar 2.8 Pemetaan *Input Output* .

Ada beberapa alasan mengapa orang menggunakan logika *fuzzy*, antara lain:

a. Konsep logika *fuzzy* mudah dimengerti. Konsep matematis yang mendasari penalaran *fuzzy* sangat sederhana dan mudah dimengerti.

b. Logika *fuzzy* sangat fleksibel.

c. Logika *fuzzy* memiliki toleransi terhadap data-data yang tidak tepat.

d. Logika *fuzzy* mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks.

e. Logika *fuzzy* dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.

f. Logika *fuzzy* dapat bekerjasama dengan teknik-teknik kendali secara konvensional.

2.7.1 Himpunan *Fuzzy*

Pada himpunan tegas *(crisp)*, nilai keanggotan suatu *item* x dalam himpunan A yang sering ditulis dengan μA[x], memiliki 2 kemungkinan yaitu:

1. Satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan.
2. Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

2.7.2 Fungsi Keanggotaan

Fungsi keanggotaan adalah kurva yang menunjukkan pemetaan titik – titik *input* data kedalam nilai keanggotaannya (derajat keanggotaan) yang memiliki interval antara 0 sampai 1.

2.7.3 Operator Dasar

Seperti halnya himpunan konvensional, ada beberapa operasi yang didefinisikan secara khusus untuk mengkombinasi dan memodifikasi himpunan *fuzzy*.

a. Operator AND

Operator ini berhubungan dengan operasi interseksi pada himpunan. α-predikat sebagai hasil operasi dengan operator AND diperoleh dengan mengambil nilai keanggotaan terkecil antar elemen pada himpunan – himpunan yang bersangkutan.

$$\mu A \cap B = \min(\mu A[x], \mu B[y]) \quad \text{.....(2.13)}$$

b. Operator OR

Operator ini berhubungan dengan operasi union pada himpunan. α-predikat sebagai hasil operasi dengan operator OR diperoleh dengan mengambil nilai keanggotaan terkecil antar elemen pada himpunan – himpunan yang bersangkutan.

$$\mu AUB = \max(\mu A[x], \mu B[y]) \quad \text{.....(2.14)}$$

c. Operator NOT

Operator ini berhubungan dengan operasi komplemen pada himpunan. α-predikat sebagai hasil operasi dengan operator NOT diperoleh dengan mengambil nilai keanggotaan terkecil antar elemen pada himpunan – himpunan yang bersangkutan.

$$\mu A' = 1-\mu A[x] \quad \text{.....(2.15)}$$

2.7.4 Penalaran Monoton

Metode ini digunakan sebagai dasar untuk teknik implikasi *fuzzy*. Jika 2 daerah *fuzzy* direalisasikan dengan implikasi sederhana sebagai berikut:

IF x is A THEN y is B

transfer function:

Y = f ((x, A), B)

Maka *system fuzzy* dapat berjalan tanpa harus melalui komposisi dan dekomposisi *fuzzy*. Nilai output dapat diestimasi secara langsung dari nilai keanggotaan yang berhubungan dengan antesedennya.

2.7.5 Fungsi Implikasi

Bentuk umum aturan yang digunakan dalam fungsi implikasi:

IF x is A THEN y is B

Dengan x dan y adalah skalar, A dan B adalah himpunan *fuzzy*. Proposisi yang mengikuti IF disebut anteseden, sedangkan proposisi yang mengikuti THEN disebut konsekuen. Secara umum, ada dua fungsi implikasi, yaitu:

1. Min (minimum), fungsi ini akan memotong *output* himpunan *fuzzy*
2. Dot *(product)*, fungsi ini akan menskala *output* himpunan *fuzzy*.

2.7.6 Metode Sugeno

Penalaran dengan metode SUGENO hampir sama dengan penalaran MAMDANI, hanya saja *output* (konsekuen) sistem tidak berupa himpunan *fuzzy*, melainkan berupa konstanta atau persamaan linear. Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985.

1. Model *Fuzzy* Sugeno Orde-Nol

Secara umum bentuk model *fuzzy* SUGENO Orde-Nol adalah:

IF (x1 is A1) • (x2 is A2) • (x3 is A3) • ..... • (xN is AN) THEN z=k

Dengan Ai adalah himpunan fuzzy ke-i sebagai anteseden, dan k adalah suatu konstanta (tegas) sebagai konsekuen.

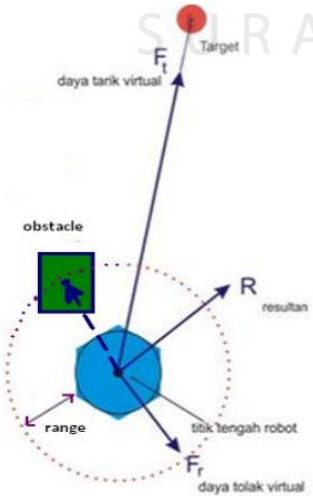
2. Model Fuzzy Sugeno Orde-Satu

Secara umum bentuk model *fuzzy* SUGENO Orde- Satu adalah:

IF (x1 is A1) • ..... • (xN is AN) THEN z =p1 * x1 + … + pN * xN + q

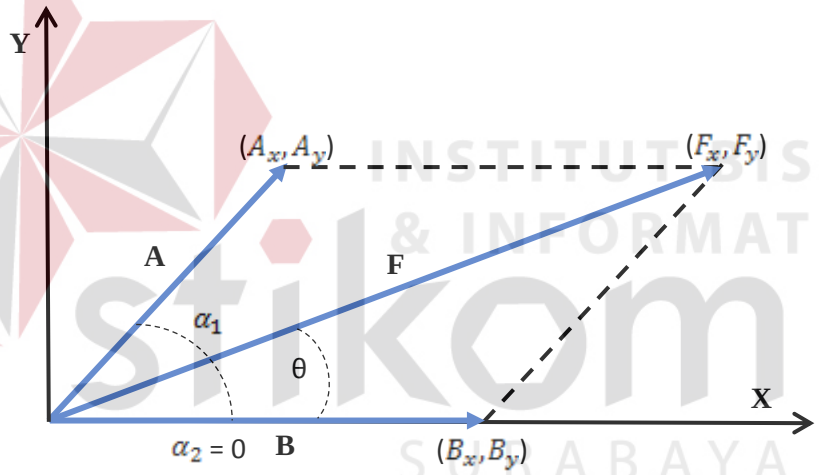
Dengan Ai adalah himpunan *fuzzy* ke-i sebagai anteseden, dan pi adalah suatu konstanta (tegas) ke-i dan q juga merupakan konstanta dalam konsekuen. Apabila komposisi aturan menggunakan metode SUGENO, maka deffuzifikasi dilakukan dengan cara mencari nilai rata-ratanya.

2.8 Metode *Virtual Force Field*



Gambar 2.9. Konsep Metode *Virtual Force Field* yang disederhanakan (Purnomo, 2011)

Metode *Virtual Force Field* (VFF) yang disederhanakan adalah berbasis vektor, bukan grid, sehingga dari metode VFF diambil konsepnya sebagian, yaitu pada perhitungan vektor. Perhitungan vektor digunakan untuk mengetahui arah yang harus dituju robot. Sedangkan untuk pemetaan obstacle, digunakan *range* tertentu untuk menentukan apakah pada area tersebut terdapat halangan atau tidak. Jika terdapat halangan maka robot akan bergerak ke arah resultan antara halangan dan target. Jika tidak, maka ia akan langsung bergerak menuju target. Untuk perhitungan besar dan arah resultan digunakan perhitungan vektor dengan metode jajar genjang dengan persamaan seperti di bawah ini.



Gambar 2.10. Metode Jajar Genjang (Jonifan, 2008)

$$A_x = A \cos \alpha_1 \quad \text{.....(2.16)}$$

$$A_y = A \sin \alpha_1 \quad \text{.....(2.17)}$$

$$B_x = B \cos \alpha_2 \quad \text{.....(2.18)}$$

$$B_y = B \sin \alpha_2 \quad \text{.....(2.19)}$$

Dimana, A_x dan A_y merupakan koordinat (x,y) dari vektor A. Sedangkan B_x dan B_y adalah koordinat (x,y) dari vektor B. Karena $\alpha_2 = 0$, maka B_x akan bernilai 1, dan B_y akan bernilai 0.

$$F_x = A_x + A_y \quad \text{.....(2.20)}$$

$$F_y = B_x + B_y \quad \text{.....(2.21)}$$

Dimana, F_x merupakan penjumlahan koordinat x kedua vektor, dan F_y merupakan penjumlahan koordinat y kedua vektor.

$$F = \sqrt{(F_x)^2 + (F_y)^2)} \quad \text{.....(2.22)}$$

Dimana F merupakan besar vektor hasil penjumlahan vektor A dan vektor B. Hasil penjumlahan tersebut memiliki arah vektor yang dapat diketahui nilainya dengan rumus sebagai berikut :

$$\tan\theta = \frac{F_y}{F_x} \quad \text{.....(2.23)}$$

Sehingga,

$$\theta = tan^{-1}\frac{F_y}{F_x} \quad \text{.....(2.24)}$$