Article Title

JOHN SMITH*

University of California

john@smith.com

March 2, 2021

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

I Pendahuluan

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

*A thank you or further information

II Metode

Penentuan koordinat tentangga dapat ditemukan dengang mengubah koordinat polar menjadi koordinat kartesian. Koordinat polar membutuhkan panjang d_a, dan sudut α. Panjang d_a adalah variable yang didapat dari sensor yang memberikan nilai jarak dari robot A ke robot B, akan tetapi untuk mendapatkan koordinat polar, pengukuran sudu α tidak tersedia. Algoritama yang ditawarkan memanfaatkan hukum *cosinus* pada segitiga untuk mendapatkan sudut tersebut.

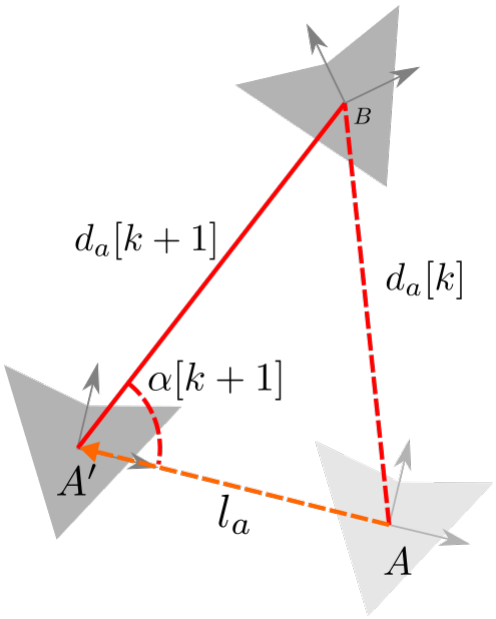

Figure 1: *Strategi Penentuan Koordinat*

Dapat diperhatikan pada gambar 1 untuk gambaran strateginya. Robot $B \in \mathcal{N}_A$, adalah tetangga dari robot A. Pertama-tama, sebelum robot A bergerak, disimpan terlebih dahulu nilai d_a, atau dinotasikan dengan $d_a[k]$ sebagai jarak sebelum bergerak. Lalu robot A berjalan secara random kesegala arah dengan jarak l_a. Disimpan kembali nilai jara d_a, atau dinotasikan dengan $d_a[k+1]$. Setalah itu dapat ditentukan sudut $\alpha[k+1]$

$$d_a[k]^2 = d_a[k+1]^2 + l_a^2 + 2d_a[k+1]l_a \cos(\alpha[k+1]) \tag{1}$$

$$\alpha[k+1] = cos^{-1}\left(\frac{l_a^2 + d_a[k+1]^2 - d_a[k]^2}{2d_a[k+1]l_a}\right) \tag{2}$$

Sebelum $\alpha[k+1]$ digunakan, jarak $d_a[k+1]$ dan $d_a[k]$ berpengaruh dalam penentuan koordinat. Sehingga diperlukan sedikit algoritma

$$\alpha_i = \begin{cases} \alpha[k+1] & ,d_a[k+1] > d_a[k] \\ 180 - \alpha[k+1] & ,d_a[k+1] < d_a[k] \end{cases}. \tag{3}$$

Strategi pada gambar 1 hanya berlaku apabila target ukur berhenti. Apabila dinotasikan koordinat (x_B^A, y_B^A) adalah koordinat relatif robot B terhadap A, maka $(\dot{x}_B^A, \dot{y}_B^A)$ adalah notasi kecepatan koordinat dari robot B. Dengan menggunakan persamaan (**??**) untuk menyelesaikan koordinat dalam keadaan robot B bergerak, yaitu mengirimkan informasi kecepatan koordinatnya ke robot A. Lalu robot A dapat mengkalkulasi koordinat relatif dengan persamaan berikut

$$\alpha[k+1] = \alpha[k] + tan^{-1}\left[\frac{\dot{x}_B^A}{\dot{y}_B^A}\right] \tag{4}$$

dimana kondisi inisial adalah $\alpha[k] = \alpha_i$ diperoleh dari hasil strategi pada persamaan (3). Dengan memanfaatkan kedua strategi tersebut dapat digunakan untuk mengkalkulasi koordinat robot B relatif terhadap robot A

$$x_B^A = \begin{bmatrix} x_B = d_a[k]\cos\alpha[k] \\ y_B = d_a[k]\sin\alpha[k] \end{bmatrix} \tag{5}$$

Dalam strategi ini akan terjadi ketidak akuratan dalam pengukuran apabila target ukur berada pada sudut 90°. Akan tetapi, Cao, dkk (2007) sudah menjelaskan mengenai kriteria posisi agent ketika dalam kondisi inisial. Yaitu semua agent tidak berada pada

kondisi sejajar secara koordinat global.

Algoritme 1: *Algoritma Cosinus*

Masukan: Integer $l_a > 0$,
$\mathcal{N}_i = getConnectionRobot()$,
Keluaran: x_i^j

if *isInisilised()* == *false* **then**
  /* inisialisasi */
  /* getRandomDirection() akan mengembalikan sudur random antara 0 - 360 */
  $dir = getRandomDirection()$
  $d_{before} = getDistanceFromSensor(\mathcal{N}_i)$
  $r = \begin{bmatrix} l_a \cos(dir) \\ l_a \sin(dir) \end{bmatrix}$
  /* Menjalankan robot hingga mencapai setpoint */
  while *isSetpointReached()* **do**
    $runRobotToSetpoint(r)$
  /* Mengambil jarak setelah robot mencapai setpoint */
  $d_{after} = getDistanceFromSensor(\mathcal{N}_i)$
  /* Mengkalkulasi sudut */
  $ang = cos^{-1}\left[\frac{l_a^2 + d_{after}^2 - d_{before}^2}{2 d_{before} l_a}\right]$
else
  /* mendapatkan infromasi state dari tetangga */
  $\begin{bmatrix} \dot{x}_B^A \\ \dot{y}_B^A \end{bmatrix} = getState()$
  $ang = \alpha[k] + tan^{-1}\left[\frac{\dot{x}_B^A}{\dot{y}_B^A}\right]$
if $d_{before} < d_{after}$ **then**
  $ang = 180 - ang$
/* Menjadikan koordinat kartesian */
return $x_i^j = \begin{bmatrix} d_{after} \cos(ang) \\ d_{after} \sin(ang) \end{bmatrix}$

III Hasil

IV Diskusi

V DAFTAR REFERENSI

DAFTAR REFERENSI

Cao, M., dkk (2007). “Controlling a triangular formation of mobile autonomous agents”. In: *2007 46th IEEE Conference on Decision and Control*, pp. 3603–3608. DOI: 10.1109/CDC.2007.4434757.