PERANCANGAN PENGENDALI ROBOT BERGERAK BERBASIS PERILAKU MENGGUNAKAN *PARTICLE SWARM FUZZY CONTROLLER*

Andi Adriansyah

Program Studi Teknik Elektro, Fakultas Teknologi Industri, Universitas Mercu Buana, Jalan Raya Meruya Selatan, Jakarta, 11650, Indonesia

E-mail: andi@mercubuana.ac.id

Abstrak

Paper ini memaparkan perancangan pengendali robot berbasis perilaku menggunakan *Fuzzy*, di mana parameter *Fuzzy* ditala secara otomatis menggunakan *Particle Swarm Optimization* (PSO) yang diistilahkan dengan *Particle Swarm Fuzzy Controller (*PSFC*)*. Suatu fungsi tertentu dirancang untuk meningkatkan performa proses pencarian PSO. Fungsi tersebut mengubah harga bobot inersia menjadi berkurang secara *sigmoid* (*Sigmoid Decreasing Inertia Weight*). Empat buah perilaku robot dirancang menggunakan PSFC. Kemudian seluruh perilaku tersebut juga dikoordinasikan menggunakan PSFC. Beberapa simulasi pengendalian pergerakan robot dan percobaan dengan robot MagellanPro telah dilakukan untuk menguji performa algoritma yang dirancang. Algoritma lain, *Genetic Fuzzy Controller* (GFC) digunakan sebagai pembanding. Dari hasil pengujian dapat dikatakan bahwa pengendali yang dirancang memiliki kemampuan yang baik untuk menyelesaikan tugasnya pada suatu lingkungan nyata.

Kata Kunci: *particle swarm optimization*, *pengendali logika fuzzy, robot berbasis perilaku, robot bergerak,*

Abstract

This paper describes the design of robots controllers based on behaviour using Fuzzy, in which the Fuzzy parameters are automatically tuned using the Particle Swarm Optimization (PSO) which is termed the Particle Swarm Fuzzy Controller (PSFC). A particular function is designed to improve the performance of PSO search process. That particular function changes the value of the inertia weight, so it's decreased in sigmoid (Sigmoid Decreasing Inertia Weight). Four types of robots behaviour are designed and coordinated using the PSFC. Some simulation of the robot movement control and experiments with the robot MagellanPro have been conducted to test the performance of the algorithm that have been designed. Another algorithm, Genetic Fuzzy Controller (GFC) is used as a comparison. From the test results, it can be said that the controllers that have been designed have a good ability to accomplish its task in a real environment.

Keywords: *behavior-based robots, fuzzy logic controllers, moving robot, particle swarm optimization*

1. Pendahuluan

Saat ini, pengendali robot bergerak berbasis perilaku (*behavior-based controller*) menjadi alternatif utama bagi penggunaan pengendali robot konvensional [1]. Namun, beberapa *problem* masih harus diselesaikan untuk dapat mengaplikaskan teknik tersebut [2]. Beberapa metode sistem kendali telah dicoba untuk menyelesaikan *problem* ini. Namun, metode sistem kendali konvensional biasanya menghadapi permasalahan komputasi yang rumit dan setiap metode hanya cocok untuk suatu tipe robot saja. Hal ini menyebabkan masih diperlukan metode pengendalian yang lebih handal dalam mekanisme komputasi dan sesuai bagi berbagai tipe robot [3].

Beberapa teknik Kecerdasan Buatan atau *Artificial Intelligence* (AI), baik secara individual maupun hibrida telah dicobakan untuk menyelesaikan *problem* robot berbasis perilaku ini, seperti logika *Fuzzy*, Jaringan Syaraf Tiruan atau Komputasi Evolusi [4]. Logika *Fuzzy* termasuk teknik yang sering diaplikasikan untuk mengimplemantasikan pengendali perilaku individu [5][6]. Bukan pekerjaan mudah untuk mendapatkan parameter yang dibutuhkan oleh sebuah pengendali logika *Fuzzy* yang handal. Oleh karena itu, diperlukan proses pelatihan dan pembelajaran (*training and learning*) untuk menala parameter-parameter tersebut.

Paper ini mempresentasikan sebuah pendekatan untuk menyelesaikan *problem* di atas

dengan menggunakan *Particle Swarm Fuzzy Controller* (PSFC). PSFC ini didasari oleh prinsip pengendali logika *Fuzzy* yang parameter logika *Fuzzy*-nya ditala secara otomatis menggunakan *Particle Swarm Optimization* (PSO).

2. Metodologi

Pengendali robot berbasis perilaku dikembangkan berdasarkan pendekatan yang diinspirasi oleh sistem makhluk hidup. Makhluk hidup beraktivitas berdasarkan modul-modul tertentu yang disebut perilaku (*behavior*), yang bekerja secara paralel untuk mencapai tujuan tertentu [1][7][8]. Struktur dasar pengendali ini terdiri dari beberapa perilaku, di mana setiap perilaku memiliki masukan yang berasal dari sensor-sensor dan menghasilkan keluaran berupa penggerak dari robot. Sebuah koordinator perilaku (*behavior coordinator*) diperlukan untuk mengkoordinasikan keluaran dari perilaku-perilaku yang aktif pada waktu yang bersamaan.

Pengendali robot berbasis perilaku dengan logika *Fuzzy* berarti menggunakan teknik logika *Fuzzy* pada setiap perilakunya [9]. Arsitektur umum pengendali ini dibentuk oleh dua blok utama, yaitu blok modul-modul pengendali perilaku dan blok koordinator yang direpresentasikan pada gambar 1.

Sebagaimana telah disebutkan di atas, untuk mendapatkan parameter pengendali logika *Fuzzy* yang handal tidaklah mudah. Selain itu, penambahan variabel pada logika *Fuzzy* berakibat penambahan aturan *Fuzzy* secara eksponensial sehingga semakin membuat rumit parameter yang diperlukan. Oleh karena itu, untuk menala parameter-parameter tersebut diperlukan proses pelatihan dan pembelajaran (*training and learning*).

Dalam dekade ini, beberapa kajian difokuskan untuk menghasilkan sistem *Fuzzy* yang memiliki kemampuan pembelajaran dan penalaan secara otomatis [10]. Contohnya sistem *Neuro-Fuzzy*, sebagai teknik yang diminati dan didukung oleh banyak peneliti [11][12]. Selain itu, terdapat pula teknik lain, seperti: sistem *Fuzzy* dengan *Reinforcement Learning* [13], *Genetic Fuzzy System* [14], *Fuzzy Genetic Programming* [15], dan lain-lain. PSFC adalah teknik baru yang akan dibahas pada *paper* ini.

Particle Swarm Optimization (PSO) adalah teknik optimasi dengan menyimulasikan perilaku sosial makhluk hidup kecil, seperti sekawanan ikan atau burung, yang bergerak sesuai dengan tujuan tertentu [16][17]. Prinsip kerja PSO adalah sebagai berikut: setiap solusi potensial, disebut dengan *partikel*, memiliki nilai terbaik (*pbest*) berdasarkan posisinya dan setiap partikel juga memiliki nilai terbaik kelompok (*gbest*) di antara nilai terbaik posisinya (*pbest*). Keseluruhan nilai terbaik didasari oleh sebuah fungsi terbaik (*fitness function, F*) untuk setiap solusi. Setiap partikel berupaya dari waktu ke waktu untuk memodifikasi posisinya menggunakan kecepatan dan posisi sesaatnya. Kecepatan untuk setiap partikel dihitung berdasarkan persamaan 1:

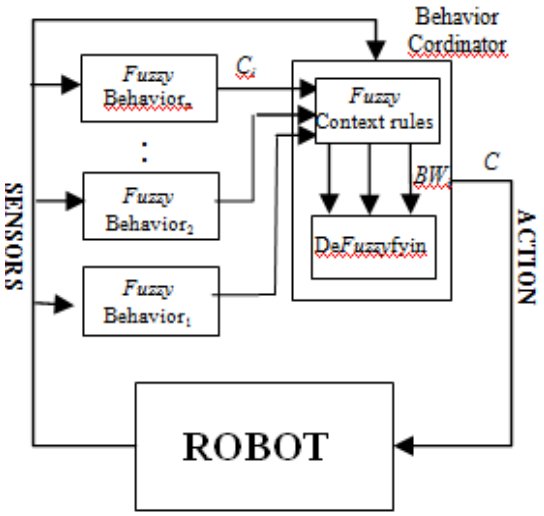
$$v_i^{k+1} = w_i * v_i^k + c_1 * rand * (pbest - s_i^k) + c_2 * rand * (gbest - s_i^k) \quad (1)$$

Di mana $v_i^{\ k}$, v_i^{k+i}, dan $s_i^{\ k}$, masing-masing adalah vektor kecepatan, vektor kecepatan waktu berikut, dan vektor posisi dari partikel *i* pada iterasi ke-*k.* Sedangkan *pbest* and *gbest* adalah posisi dengan nilai terbaik partikel *i* dan nilai terbaik kelompoknya serta c_j dan w_j masing-masing adalah koefisien bobot dan fungsi inersia untuk kecepatan bagi setiap partikel *i*. Setelah itu, posisi partikel berikutnya dihitung berdasarkan persamaan 2, yaitu:

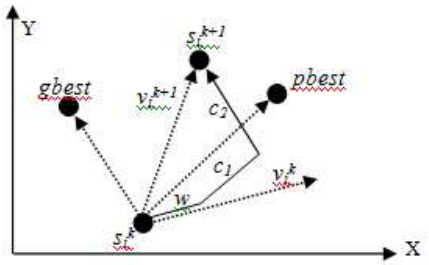
$$s_i^{k+1} = s_i^k + v_i^{k+1} \quad (2)$$

Proses pemutakhiran partikel berdasarkan persamaan 1 dan 2 diilustrasikan pada gambar 2. Didapatkan kesimpulan dari Algoritma PSO untuk mendapatkan vektor posisi terbaik menggunakan sejumlah partikel n, yakni vektor posisi awal *S*[*n*] dan vektor kecepatan *V*[*n*] dihasilkan nilainya secara acak . Selain itu vektor kecepatan v_i^{k+i} partikel *i* dihitung berdasarkan persamaan (1), sementara vektor posisi s_i^{k+1} partikel *i* dimutakhirkan berdasarkan persamaan (2). Kemudian jika nilai F(s_i^k) lebih baik dari F($pbest_i$), vektor posisi s_i^k di-*set* menjadi *pbest*. Namun, jika nilai F($pbest_i$) lebih baik dari F(*gbest*), maka vektor posisi *gbest* di-*set* menjadi *pbest*. Terakhir jika iterasi yang telah dilakukan telah mencapai harga tertentu, algoritma dihentikan. Jika tidak, kembali mengerjakan tahap 2.

Untuk menciptakan proses pencarian partikel terbaik, diperkenalkan konsep pembobotan inersia (*inertia weight, w*). Pemanfaatan konsep pembobotan inersia pertama kali memiliki harga *w* yang dihasilkan secara linier menurun (*decreased linearly*) dari 0.9 hingga 0.4. Pemanfaatan konsep ini dapat meningkatkan performansi optimasi selama proses pencarian berlangsung [18].



Gambar 1. Pengendali robot berbasis perilaku dengan logika *Fuzzy*.

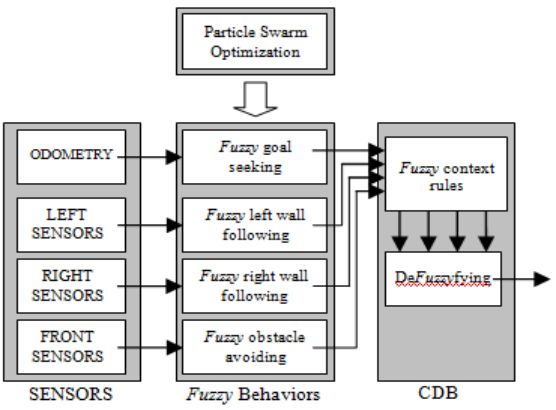


Gambar 2. Pemutakhiran kecepatan dan posisi pada PSO.

Paper ini menawarkan metode yang lain, yaitu perubahan harga *w* secara menurun berdasarkan fungsi *sigmoid* (*Sigmoid Decreasing Inertia Weight,* SDIW). Pengimplementasian SDIW menghasilkan proses pencarian yang kompromistis antara eksplorasi dan eksplorasi. Nilai bobot inersia yang besar akan menyebabkan proses pencarian global pada awal proses. Sedangkan nilai bobot inersia yang kecil mengakibatkan proses pencarian *local* pada bagian akhir proses. Dengan demikian, metode ini menghasilkan keseimbangan antara pencarian *global* dan *local* sehingga menyebabkan pencarian yang terbaik.

Particle Swarm Fuzzy Controller (PSFC) adalah sistem *Fuzzy* yang ditambahkan dengan kemampuan pembelajaran menggunakan PSO. PSO diimplementasikan di sistem *Fuzzy* untuk melakukan pencarian parameter sistem *Fuzzy* yang handal untuk suatu *problem* tertentu. Pada *paper* ini, PSFC diterapkan secara dua tahap. Pada tahap pertama, PSO melakukan pencarian aturan sistem *Fuzzy* (*Fuzzy rule base*) berdasarkan nilai fungsi keanggotaan (*membership function*) yang ditentukan terlebih dahulu. Pada tahap kedua, pencarian dilakukan untuk mendapatkan nilai fungsi keanggotaan berdasarkan aturan sistem *Fuzzy* yang dihasilkan. Dengan metode ini, keseluruhan parameter sistem *Fuzzy* yang handal dapat dihasilkan secara otomatis.

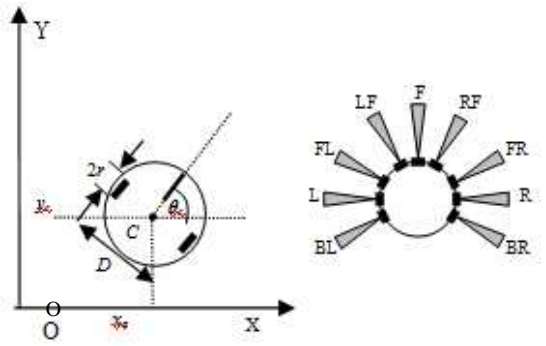
Arsitektur pengendali robot bergerak berbasis perilaku yang dirancang pada *paper* ini memiliki blok diagram seperti yang ditampilkan pada gambar 3. Robot terdiri dari empat perilaku, yaitu perilaku menuju target (*goal seeking*), pengikut dinding kiri (*left wall following*), pengikut dinding kanan (*right wall following*), dan penghindar rintangan (*obstacle avoiding*). Setiap perilaku dikendalikan dengan pengendali logika *Fuzzy* yang memproses masukan dari masing-masing sensor untuk menghasilkan keluaran tertentu sesuai dengan perilakunya. Keseluruhan parameter sistem *Fuzzy* dari setiap perilaku ditala menggunakan PSO. Kemudian semua perilaku tersebut dikoordinasikan untuk mendapatkan aksi akhir terbaik bagi berbagai situasi dengan menggunakan Penggabungan Bergantung Konteks (*Context Dependent Blending*, CDB).



Gambar 3. Rancangan arsitektur pengendali robot bergerak.

Perancangan model robot bergerak akan diujikan pada sebuah model robot. Model robot didasari oleh robot bergerak MagellanPro yang berbentuk silindris. Robot ini digerakkan oleh dua buah motor DC secara independen. Robot memiliki dimensi sebagai berikut: D = 40.6 cm, H = 25.4 cm, r = 5.7 cm, dan W = 36 cm, yaitu D adalah diameter, H adalah tinggi, r adalah jari-jari roda, dan W adalah jarak antara roda. Robot tersebut diilustrasikan pada gambar 4(a) dengan konfigurasi sensor ultrasonik seperti pada gambar 4(b).

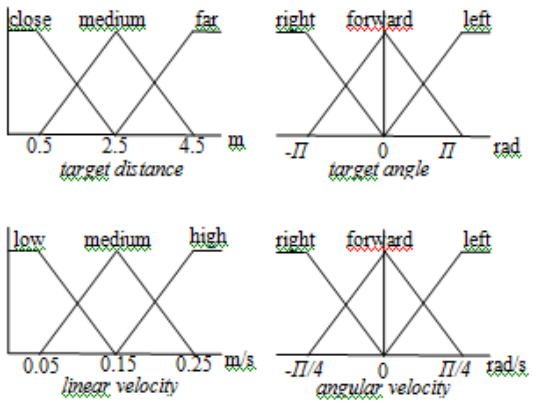
Robot diposisikan pada suatu bidang berkoordinat Cartesius $\{X,O,Y\}$ dengan tiga derajat kebebasan dan direpresentasikan sebagai titik pusat $p_c = (x_c, y_c, \theta_c)$, di mana (x_c, y_c) adalah posisi spasial robot dan θ_c adalah sudut robot terhadap sumbu horizontal.



Gambar 4. Model robot bergerak: (a) dimensi dan posisi dan (b) konfigurasi sensor.

Keseluruhan sistem *Fuzzy* untuk setiap perilaku mempunyai struktur yang sama, yaitu memiliki fungsi keanggotaan berbentuk trapesium dan segitiga. Setiap perilaku individu mendapatkan masukan dari sinyal sensor $(x_1,x_2,\ldots,x_n)$, dan memiliki aksi pengendalian (y_1,y_2) yang dihasilkan dari keluarannya.

Pada umumnya, perilaku yang dirancang memiliki sistem *Fuzzy* yang terdiri dari tiga masukan dan dua keluaran. Hanya pada perilaku menuju target sistem memiliki dua masukan, yaitu *target distance* (d) dan *target angle* (θ). Pada perilaku pengikut dinding kiri, masukannya adalah *front left distance* (FL), *left distance* (L), dan *back left distance* (BL). Pada perilaku pengikut dinding kanan masukannya adalah *front right distance* (FR), *right distance* (R), dan *back right distance* (BR). Pada perilaku penghindar rintangan adalah *left front distance* (LF), *front distance* (F), dan *right front distance* (RF). Setiap masukan memiliki term bahasa sebagai berikut: CLOSE, MEDIUM, dan FAR untuk jarak serta RIGHT, FORWARD, dan LEFT untuk sudut. Sementara itu, semua perilaku memiliki keluaran yang sama, yaitu kecepatan linear, (v), dan kecepatan angular, (ω), dengan term bahasa, masing-masing: LOW, MEDIUM, dan HIGH serta LEFT, FORWARD, dan RIGHT. Sebagai contoh fungsi keanggotaan *Fuzzy* untuk perilaku menuju target diperlihatkan pada gambar 5.



Gambar 5. Fungsi keanggotaan dari (a) jarak masukan, (b) sudut masukan, dan (c) keluaran kecepatan linier serta kecepatan angular.

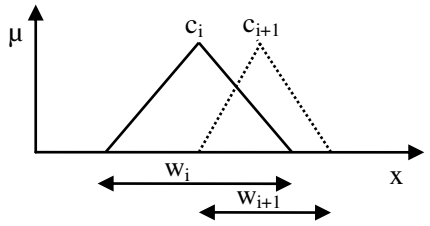
Setelah itu, aturan *Fuzzy* tertentu dirancang berbasiskan pendekatan aturan konteks *Fuzzy* fleksibel (*flexible Fuzzy context rule*, FFCR). FFCR digunakan pada *paper* ini untuk menghasilkan koordinasi yang strategis bagi interaksi antara perilaku yang sesuai untuk berbagai keadaan. FFCR dirancang berasaskan aturan *Fuzzy* dengan struktur konjungsi dan negasi. PSO yang sama juga akan diimplementasikan untuk mendapatkan aturan *Fuzzy* yang efektif.

PSO bekerja menggunakan parameter yang dikodekan sebagai partikel, oleh karena itu semua parameter sistem *Fuzzy* yang dibutuhkan harus dikodekan dalam rangkaian kode tertentu. Pertama, untuk mencari aturan *Fuzzy* terbaik, setiap aturan dikodekan dalam kode bilangan *integer* yang menggambarkan sejumlah term bahasa keluaran fungsi keanggotaan *Fuzzy*. Misalnya, kode '1', '2', dan '3' masing-masing untuk kode term bahasa LOW, MEDIUM, dan HIGH bagi keluaran kecepatan linear, (v), serta RIGHT, FORWARD, dan LEFT bagi kecepatan angular, (ω). Pada tahap kedua, untuk menala fungsi keanggotaan sistem *Fuzzy* digunakan persamaan 3:

$$C_x = C_x + k_i W_x = W_x + j_i \tag{3}$$

Di mana k_i dan j_i adalah koefisien yang ditala. C_x dan W_x adalah harga pusat dan lebar untuk masing-masing fungsi keanggotaan. Jadi

harga k_i akan mempengaruhi pergeseran fungsi keanggotan ke kanan atau ke kiri. Sedangkan harga j_i mempengaruhi kelebaran fungsinya sebagaimana diperlihatkan pada gambar 6. Setelah itu kode-kode tersebut dirangkai untuk menjadi sebuah partikel lengkap.



Gambar 6. Prinsip penalaan fungsi keanggotaan.

Proses pencarian PSO kemudian dimulai dengan menentukan populasi awal secara acak. Lalu seluruh populasi dievaluasi dan disesuaikan posisinya berdasarkan fungsi terbaik untuk menentukan *pbest* dan *gbest*. Fungsi terbaik yang dirancang berturut-turut untuk perilaku menuju target, pengikut dinding (kiri dan kanan), dan penghindar rintangan adalah sebagai berikut:

$$f_{goal} = \sum_{i=0}^{I} \sum_{k=0}^{K} (100e_{\theta}^{2}(k) + e_{d}^{2}(k) + 100 / v(k)) \quad (4)$$

$$f_{wall} = \sum_{i=0}^{I} \sum_{k=0}^{K} (100e_{d}^{2}(k) + 100\omega^{2}(k) + 0.1 / v(k)) \quad (5)$$

$$f_{obs} = \sum_{i=0}^{I} \sum_{k=0}^{K} (100\omega^{2}(k) + 0.5 / v(k) + 100c(k)) \quad (6)$$

I adalah jumlah total posisi awal robot, K adalah jumlah tahapan simulasi untuk setiap posisi awal robot, e_θ adalah kesalahan sudut, e_d adalah jarak, $v(k)$, dan $\omega(k)$ adalah kecepatan linear dan kecepatan angular pada saat k, serta c adalah konstanta untuk mengecek apakah terjadi tabrakan terhadap rintangan atau tidak. Secara umum, tujuan dari fungsi-fungsi terbaik tersebut adalah memastikan bahwa robot bergerak dengan kecepatan tercepat, arah yang tepat, berada pada jarak tertentu pada dinding dalam rangka menuju suatu target tertentu.

Setelah itu, vektor kecepatan dan vektor posisi yang baru dihitung berdasarkan persamaan-persamaan yang telah dibahas sebelumnya. Prosedur ini dilakukan hingga kondisi yang diharapkan tercapai untuk setiap perilaku yang dirancang.

Terakhir, fungsi terbaik untuk aturan konteks ditentukan sesuai dengan persamaan 7:

$$f_{goal} = \sum_{i=0}^{I} (c_1.Time + c_2.Way + c_3.Coll + c_4.DeltaWallSq) \quad (7)$$

I adalah jumlah iterasi yang dihubungkan dengan jumlah posisi target, *Time* adalah persentase dari jumlah tahap simulasi yang telah dilakukan terhadap waktu yang disediakan, *Way* adalah persentase dari jarak yang tersisa dari posisi awal ke posisi target selama robot berjalan, *Coll* adalah jumlah tabrakan terhadap rintangan yang terjadi, dan *DeltaWallSq* adalah jumlah kuadrat dari jarak antara dinding kiri dan kanan. Fungsi terbaik ini dirancang agar robot dapat bergerak dalam waktu eksekusi tercepat, penunaian tugas yang tuntas, tanpa tabrakan terhadap rintangan, dan menjaga jarak tertentu kepada dinding.

3. Hasil dan Pembahasan

Beberapa percobaan telah dilakukan dalam rangka menguji validitas dan mendemonstrasikan performa teknik yang dirancang. Beberapa percobaan menyimulasikan teknik yang dirancang menggunakan Komputer PC Pentium 2.20 GHz dan memori RAM 512 MB. Bahasa pemrograman MATLAB Versi 6.5 Release 13 digunakan untuk menyimulasikan ini. Hasil pengujian terdahulu menggunakan *Genetic Algorithm* (GA) digunakan sebagai pembanding.

Secara umum, setiap proses PSO dijalankan dengan jumlah populasi sebanyak 40 buah dan bekerja dengan 100 generasi (iterasi). Fungsi *sigmoid* simetris bagi SDIW digunakan dengan $n = 0.5$ dan harga bobot inersia *w_start* dan *w_end* masing-masing adalah 0.9 dan 0.4. Koefisien c_1 and c_2 diberi nilai yang sama yaitu 2.0. Persamaan 4 hingga persamaan 6 digunakan sebagai fungsi terbaik untuk setiap perilaku. Untuk mendapatkan hasil terbaik, setiap proses PSFC dilakukan sebanyak 10 kali pengulangan.

Sebuah lingkungan dirancang sebagai tempat pengujian perilaku. Dimensi lingkungan tersebut adalah 10x10m dan dilingkupi oleh dinding sebagai pembatas. Sebuah kotak kecil diasumsikan sebagai target. Beberapa titik yang berurutan dianggap sebagai objek, baik dinding atau rintangan. Beberapa posisi awal robot akan dicobakan untuk mengetahui keandalan algoritma ini. Waktu diukur dengan banyaknya iterasi t. Di mana diasumsikan bahwa setiap t adalah 1 detik.

Pada setiap perliku diinvestigasi perbandingan penggunaan PSO dan GA. Tabel I menampilkan harga fungsi terbaik dari setiap

perilaku tersebut. Semakin kecil harga fungsi terbaik mengindikasikan performa terbaiknya. Dari tabel I dapat dinyatakan bahwa harga-harga fungsi terbaik menggunakan PSO memiliki nilai yang lebih baik dari menggunakan GA. Selain itu, beberapa pergerakan robot diujicobakan untuk melihat performa setiap perilaku. Beberapa posisi awal ditentukan untuk dapat menganalisis secara komprehensif. Orientasi pergerakan robot diperlihatkan dalam bentuk garis kecil pada robot, sedangkan kerapatan antara robot merepresentasikan kecepatan pergerakan robot. Pengendali perilaku menggunakan aturan *Fuzzy* manual (FLC) dan menggunakan GA (disebut GFC) dijadikan sebagai perbandingan.

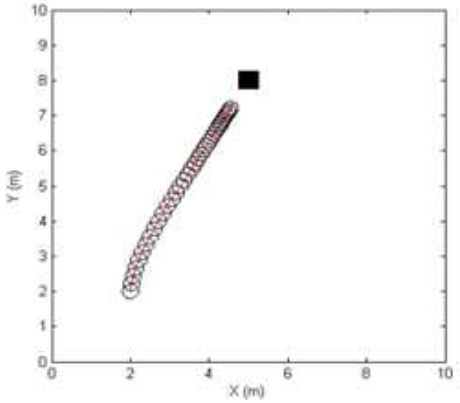
Sebagai contoh diperlihatkan analisis perilaku menuju target. Untuk menunjukkan efektifitas pergerakan robot mencapai target secara efektif diperlihatkan variabel jarak target (d), dan sudut target (δ). Pada percobaan ini, target digambarkan dalam bentuk kotak kecil. Simulasi pergerakan robot diperlihatkan pada gambar 7 dan 8.

Berdasarkan gambar 7, dapat dinyatakan bahwa ketiga algoritma menampilkan performa yang baik. Gambar 8 menggambarkan respon waktu sistem yang lebih jelas menggambarkan masing-masing performanya. Walaupun ketiga algoritma tersebut efektif untuk menghasilkan arah sehingga memiliki respon waktu sudut target yang sama tetapi terjadi perbedaan pada respon waktu jarak target. Pengendali menggunakan PSFC memiliki hasil terbaik dengan mencapai target dengan waktu 28 detik, sementara menggunakan GFC mencapai target dengan waktu 35 detik. Lebih dari itu setelah 35 detik robot belum mencapai target ketika menggunakan FLC.

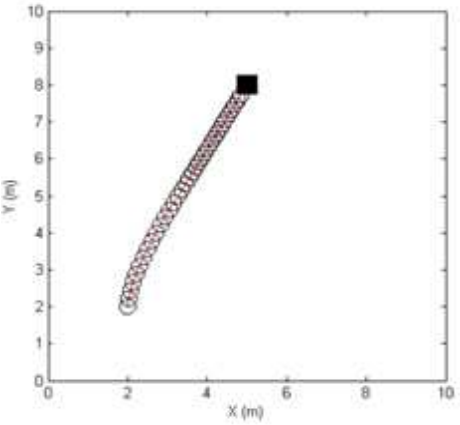
TABEL I
PERBANDINGAN HARGA FUNGSI TERBAIK

Proses / Perilaku	GA	PSO
Menuju Target	1.2097×10^{6}	1.1937×10^{6}
Pengikut Dinding	81.4125	68.4668
Penghindar Rintangan	2.0438×10^{4}	2.0158×10^{4}

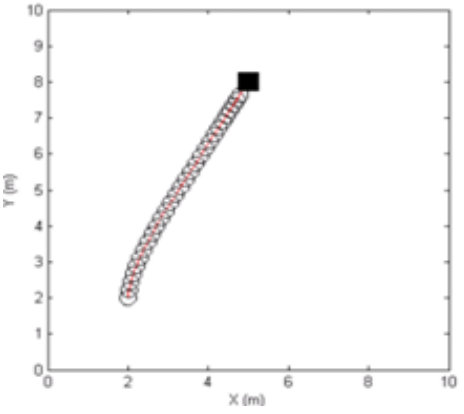
Dalam rangka mengkaji lebih lanjut, algoritma yang dirancang diujikan secara simulasi dan pergerakan robot sebenarnya pada lingkungan aktual. Percobaan ini didasari oleh pekerjaan Hoffmann. Robot bergerak harus mencapai posisi target yang diletakkan di sebelah kanan dari rintangan melalui suatu koridor tertentu. Gambar 9(a) menunjukkan hasil simulasi berdasarkan algoritma dari Hoffmann. Simulasi pergerakan menggunakan robot dan percobaan dengan robot sebenarnya, MagellanPro, ditampilkan masing-masing pada gambar 9(b) dan 9(c). Dari gambar tersebut dapat dinyatakan bahwa robot MagellanPro dengan algoritma PSFC memiliki kemampuan untuk menyelesaikan tugasnya dengan karakteristik yang sama simulasinya dengan pekerjaan peneliti terdahulu.



(a)

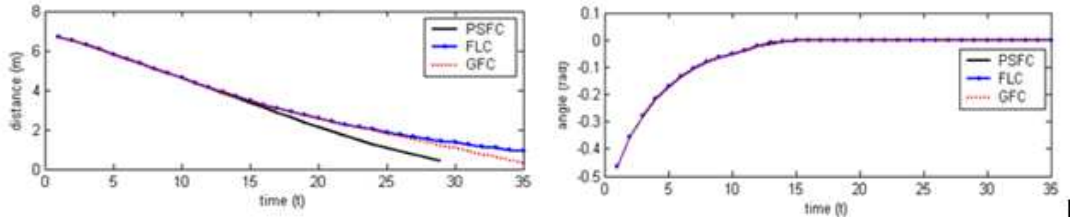


(b)

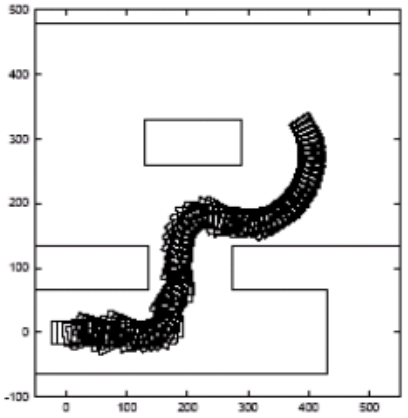


(c)

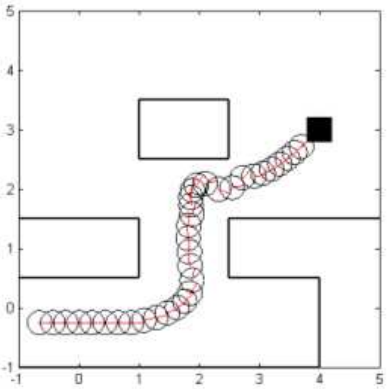
Gambar 7. Pergerakan robot untuk perilaku menuju target dengan posisi awal (2, 2, π/2) menggunakan: (a) FLC, (b) GFC, dan (c) PSFC.



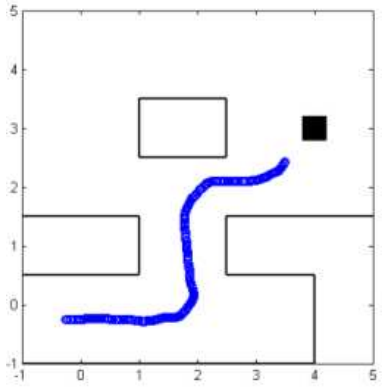
Gambar 8. Respon waktu untuk perilaku menuju target dengan posisi awal (2, 2, π/2) menggunakan FLC, GFC, dan PSFC.



(a)

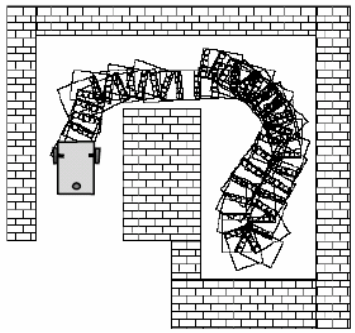


(b)

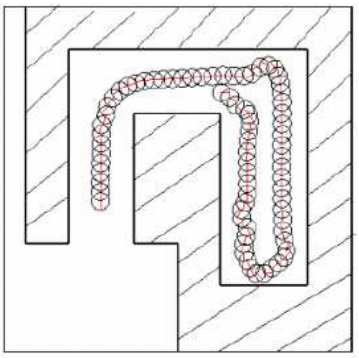


(c)

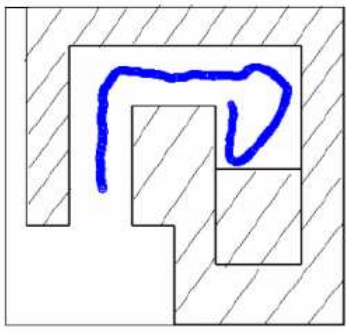
Gambar 9. Pergerakan robot pada bidang Hoffmann: (a)Hoffmann's, (b) simulasi, dan (c) robot MagellanPro.



(a)



(b)



(c)

Gambar 10. Pergerakan robot pada bidang Hoffmann yang lain: (a)Hoffmann's, (b) simulasi, dan (c) robot MagellanPro.

Percobaan berikutnya menggunakan pekerjaan Hoffmann pada lingkungan aktual yang lain. Robot harus bergerak pada koridor yang sempit, menghadapi jalan tertutup, dan keluar darinya untuk mencapai target yang berlawanan arah dari posisi awal. Dalam percobaan pengendalian robot, bidang ini merupakan keadaan yang sulit bagi robot untuk dapat keluar dari jalan tertutup. Selain itu robot harus menuju target yang arahnya berlawanan dengan posisi awal.

Sebagaimana percobaan sebelumnya, performa dari pengendali yang diimplementasikan pada MagellanPro robot dibandingkan dengan hasil dari Hoffmann dan hasil simulasi seperti ditampilkan pada gambar 10. Terlihat bahwa robot dapat bergerak dengan baik untuk menyelesaikan tugasnya. Pergerakan MagellanPro robot hampir sama dengan pergerakan simulasi dan hasil dari Hoffmann.

4. Kesimpulan

Paper ini telah mempresentasikan proses perancangan *Particle Swarm Fuzzy Controller* (*PSFC*), di mana pengendali merupakan pengendali logika *Fuzzy* yang parameternya ditala secara otomatis menggunakan *PSO*. Untuk meningkatkan performa pencarian dilakukan modifikasi *PSO* dengan menginisiasi perubahan bobot inersia menurun secara *sigmoid* (*SDIW*). Pengendali ini kemudian diimplementasikan untuk menghasilkan aksi pengendalian perilaku robot bergerak, baik perilaku individu ataupun koordinasi perilaku. Berdasarkan beberapa simulasi yang dilakukan, algoritma perancangan ini mampu membuat robot bergerak berdasarkan perilaku yang dirancang dan mengkoordinasikannya sesuai dengan kondisi yang dihadapi secara efektif. Setelah diimplementasikan ke dalam robot MagellanPro, secara umum dapat dikatakan bahwa pengendali yang dirancang memiliki kemampuan yang baik, sehingga robot dapat menyelesaikan tugas di lingkungan sebenarnya.

Referensi

[1] R.C. Arkin, *Behavior-based Robotics*, The MIT Press, Massachussets, 1998.

[2] M. Carreras, J. Batle, P. Ridao, & G.N. Roberts, "An Overview on Behavior-based Methods for AUV Control" *In Proceeding of Conference on Manufacturing and Control of Marine Crafts,* pp. 1-6, 2000.

[3] D.P. Pratihar, K. Deb, & A. Ghosh, "Fuzzy-Genetic Algorithm and Mobile Robot Navigation among Static Obstacles" *In Proceeding of The Congress of Evolutionary Computation*, pp. 327-334, 1999.

[4] F. Hoffmann & G. Pfister, "Evolutionary Design of a Fuzzy Knowledge Base for a Mobile Robot," *International Journal of Approximate Reasoning*, vol. 17, pp. 447-469, 1997.

[5] A. Saffiotti, "The Uses of Fuzzy Logic in Autonomous Robot Navigation," *Soft Computing*, vol. 1, pp. 180-197, 1997.

[6] E. Aguiree & A. Gonzales, "Fuzzy Behaviors for Mobile Robot Navigation : Design, Coordination and Fusion," *International Journal of Approximate Reasoning*, vol 25, pp. 255-289, 2000.

[7] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot, " *IEEE Journal of Robotics and Automation*, vol. 2, pp. 14-23, 1986.

[8] M.J. Mataric, "Behavior-based Control: Examples from Navigation, Learning, and Group," *Journal on Experimental and Theoretical Artificial Intelligence, Special Issue on Software Architecture for Physical Agents*, vol. 9, pp. 1-15, 1997.

[9] A. Saffiotti, E.H. Ruspini, & K. Konolige, In: Practical Application of Fuzzy Technologies, H-J. Zimmermann (Ed.), *Using Fuzzy Logic for Mobile Robot Control*, Kluwer Academic Publisher, MA, pp. 185-206, 1999.

[10] F. Hoffmann, "An Overview on Soft Computing in Behavior Based Robotics" *In Proceeding of International Fuzzy System Association World Congress IFSA* , pp. 544-551, 2003.

[11] S. Parasuraman, V. Ganapathy, & B. Shrinzadeh, "Fuzzy Decision Mechanism Combined with Neuro-fuzzy Controller for Behavior-based Robot Navigaton" *In the 29th Annual Conference of the IEEE Industrial Electronics Society,* pp. 2410-2416, 2003.

[12] P. Rusu, E.M. Petriu, T.E. Whalen, A. Cornell, & H.J.W. Spoelder, "Behavior-based Neuro-fuzzy Controller for Mobile Robot Navigation," *IEEE Transaction on Instrumentation and Measurement*, vol. 52, pp. 1335-1340, 2003.

[13] A. Bonarini, G. Invernizzi, T. Halva, & M. Matteucci, "An Architecture to Coordinate Fuzzy Behaviors to Control an Autonomous Robot," *Journal Fuzzy Sets and Systems*, vol. 134, pp. 101-115, 2003.

[14] H. Hagras, V. Callaghan, & M. Colley, "Outdoor Mobile Robot Learning and Adaptation," *IEEE Robotics & Automation Magazines*, vol. 8, pp. 53-69, September 2001.

[15] E.W. Tunstel, M.A.A. de Oliveira, & S. Berman, "Fuzzy Behavior Hierarchies for Multi-robot Control," *International Journal of Intelligent Systems*, vol. 17, pp. 499-470, 2002.

[16] J. Kennedy & R.C. Eberhart, "Particle Swarm Optimization" *In Proceeding of the 1995 Int'l Conference on Neural Networks*, pp. 1942-1948, 1995.

[17] R.C. Eberhart & Y. Shi, "Particle Swarm Optimization: Developments, Applications and Resources" *In Proceeding of the IEEE Congress on Evolutionary Computation*, pp. 81-86, 2001.

[18] R.C. Eberhart & Y. Shi, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization" *In Proceeding of the Congress on Evolutional Computation*, pp. 84-88, 2000.