TESIS – EE185401

PEMBENTUKAN FORMASI *PARTICLE SWARM* MENGGUNAKAN *ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (ANN-SOM) DENGAN STRATEGI 2 TINGKAT

BAYU FANDIDARMA
07111650022001

DOSEN PEMBIMBING
Prof. Dr. Ir. Achmad Jazidie, M.Eng.
Ir. Rusdhianto Effendi AK., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

TESIS – EE185401

PEMBENTUKAN FORMASI *PARTICLE SWARM* MENGGUNAKAN *ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (ANN-SOM) DENGAN STRATEGI 2 TINGKAT

BAYU FANDIDARMA
07111650022001

DOSEN PEMBIMBING
Prof. Dr. Ir. Achmad Jazidie, M.Eng.
Ir. Rusdhianto Effendi AK., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T.)
di
Institut Teknologi Sepuluh Nopember

oleh:

Bayu Fandidarma
NRP. 07111650022001

Tanggal Ujian : 28 Desember 2018
Periode Wisuda : Maret 2019

Disetujui oleh:

1. Prof. Dr. Ir. Achmad Jazidie, M.Eng. (Pembimbing I)
   NIP: 195902191986101001

2. Ir. Rusdhianto Effendi AK., MT. (Pembimbing II)
   NIP: 195704241985021001

3. Prof. Ir. Abdullah Alkaff, M.Sc., Ph.D. (Penguji)
   NIP: 195501231980031002

4. Dr. Trihastuti Agustinah, ST., MT. (Penguji)
   NIP: 196808121994032001

Dekan Fakultas Teknologi Elektro

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

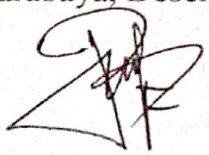
Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**PEMBENTUKAN FORMASI *PARTICLE SWARM* MENGGUNAKAN *ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (ANN-SOM) DENGAN STRATEGI 2 TINGKAT**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Desember 2018



Bayu Fandidarma

NRP. 07111650022001

Halaman ini sengaja dikosongkan

PEMBENTUKAN FORMASI *PARTICLE SWARM* MENGGUNAKAN *ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (ANN-SOM) DENGAN STRATEGI 2 TINGKAT

Nama mahasiswa : BAYU FANDIDARMA
NRP : 07111650022001
Pembimbing : 1. Prof. Dr. Ir. Achmad Jazidie, M.Eng.
2. Ir. Rusdhianto Effendi AK, MT.

ABSTRAK

Permasalahan yang diangkat dalam aplikasi *particle swarm* (agen) adalah pembentukan formasi yang terdefinisi dengan jelas pada area sekitar centroid. Salah satu metode yang pernah digunakan adalah *ANN-Self Organizing Map* (*ANN-SOM*). Namun, metode tersebut memiliki kelemahan berupa tidak mampunya membentuk formasi dengan baik terutama pada formasi *pipe obstruction*.

Pada penelitian ini, metode yang digunakan adalah *ANN-SOM* dengan menggunakan strategi pembentukan formasi dua tingkat yaitu strategi *offline* dan *online*. Posisi awal semua agen diatur sesuai dengan aturan tertentu. Bentuk formasi tujuan sudah ditentukan dan diketahui posisi akhir masing-masing agen. *ANN-SOM* akan memberikan koreksi gerak pada tiap agen. Strategi *offline* dilakukan untuk merencanakan jalur (*path planning*) dan posisi akhir dari tiap agen berdasarkan pada kriteria akumulasi jarak tempuh minimum dari semua agen. Lalu strategi *online* dijalankan dengan mengikuti jalur (*path tracking*) dan posisi akhir yang telah direncanakan. Tiap agen akan memiliki algoritma untuk menghindari tabrakan antar agen dan algoritma untuk tetap bergerak pada jalur yang telah ditentukan. Semua algoritma berjalan sampai semua agen membentuk formasi sesuai dengan posisinya masing-masing.

Eksperimen dilakukan dengan menggunakan beragam kombinasi formasi yang diinginkan dalam ruang 2D yang diberikan dan membuktian keunggulan dari metode yang diusulkan. Dari eksperimen ini memberikan hasil yang menunjukkan bahwa agen berhasil membentuk formasi sesuai dengan yang diinginkan dengan total jarak yang ditempuh oleh seluruh agen adalah minimum yaitu 73.9879.

Kata kunci: *ANN-SOM*, Pembentukan Formasi, *Path Planning*, *Path Tracking*, Jarak Minimum

Halaman ini sengaja dikosongkan

FORMING FORMATION OF PARTICLE SWARM USING ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP (ANN-SOM) WITH 2-LEVELED STRATEGY

By : BAYU FANDIDARMA
Student Identity Number : 07111650022001
Supervisor(s) : 1. Prof. Dr. Ir. Achmad Jazidie, M.Eng.
2. Ir. Rusdhianto Effendi AK, MT.

ABSTRACT

The problem raised in the application of particle swarm (agent) is forming a clearly defined formation in the area around the centroid. One method that has been used is Artificial Neural Network Self Organizing Map (ANN-SOM). However, this method has the disadvantage of being inadequate on forming formation, especially in pipe obstruction formations.

This research proposed a new method of forming formation using ANN-SOM with 2-leveled strategy, that is offline and online strategy. Initial position of all agents is regulated according to certain rules. The shape of the destination formation has been determined and the final position of each agent is known. ANN-SOM will provide motion correction for each agent. An offline strategy is done to plan the path (path planning) and the final position from each agent based on the minimum distance accumulation criteria of all agents. Then online strategy is executed by following the path (path tracking) and the final position that has been planned.

Experiments by simulation are carried out using various combinations of desired formations in a given 2D space and proving the superiority of the proposed method. The results indicate that the agents successfully formed the formation as desired with the total distance traveled by all agents is minimum that is 73.9879.

Keywords: *ANN-SOM*, Forming Formation, *Path Planning*, *Path Tracking*, Shortest Path

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Dengan segala rasa tunduk kehadirat Allah *subhanahuwata'ala*, diiringi rasa syukur atas seluruh rahmat dan ridha-Nya serta shalawat dan salam semoga senantiasa tercurah kepada junjungan Nabi besar kita, Nabi Muhammad *sholallaahu'alaihi wasallam*, sehingga penulis dapat menyelesaikan sebuah tesis dengan judul **"PEMBENTUKAN FORMASI *PARTICLE SWARM* MENGGUNAKAN *ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (ANN-SOM) DENGAN STRATEGI 2 TINGKAT"** sebagai salah satu syarat untuk menyelesaikan program Magister pada bidang keahlian Teknik Sistem Pengaturan di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember, Surabaya.

Penulis menyadari bahwa dalam penyelesaian tesis ini tak lepas dari bimbingan, dorongan, dan bantuan dari banyak pihak. Oleh karena itu, penulis menghaturkan banyak terima kasih kepada:

1. Prof. Dr. Ir. Achmad Jazidie, M.Eng. dan Ir. Rusdhianto Effendi AK., MT. selaku pembimbing tesis atas segala bimbingan dan pengarahan selama penyusunan tesis ini.
2. Semua dosen yang telah berbagi ilmu dan juga motivasi. Memberikan yang terbaik bagi penulis selama belajar dan mengembangkan diri di kampus.
3. Seluruh staf dan karyawan yang telah membantu mempermudah proses administrasi selama penulis berada di kampus.
4. Teman seperjuangan dalam menjalani penyusunan tesis; Mas Yoan, Mas Mamat dan Mas Arga. Yang telah bahu-membahu dan saling bertukar pikiran selama penyelesaian tesis ini.
5. Rekan satu angkatan perkuliahan; Tami dan Maskhur.
6. Teman-teman yang telah menemani beraktifitas di lab A202; Mas Nugraha, Mas Fahmi, Mas Dirman, Mas Khoiron, Widanis, Asti dan Mas Aceng.
7. Rekan-rekan satu bidang keahlian yang telah membersamai penulis dalam berbagai aktivitas sehingga penulis berhasil menuntaskan karya ini.

8. Bapak, ibu dan adik-adik di rumah, yang senantiasa memberikan doa dan kepercayaan yang tinggi, sehingga penulis bisa terus bersemangat dalam menjawab setiap tantangan dalam meraih mimpi besar di masa depan.
9. Semua pihak yang tidak dapat disebutkan satu persatu yang telah memberikan doa dan dukungan selama penulis menempuh pendidikan di Institut Teknologi Sepuluh Nopember hingga penyelesaian tesis ini.

Semoga Allah *subhanahuwata'ala* memberikan balasan kebaikan kepada semua pihak tersebut dan sekali lagi penulis mengucapkan terima kasih yang sebesar-besarnya.

Dalam penulisan tesis ini, penulis menyadari bahwa karya ini masih jauh dari kata sempurna. Oleh karena itu, penulis memohon maaf atas segala kekurangannya. Saran dan kritik yang bersifat membangun akan selalu penulis terima dengan harapan semoga karya ini dapat berguna bagi pembaca, dan dapat dilanjutkan untuk memperoleh hasil yang bermanfaat di kemudian hari.

Surabaya, 12 Desember 2018
Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Aplikasi *swarm* memiliki potensi yang besar untuk diteliti. Tiap agen dari *swarm* dapat berkolaborasi dengan agen *swarm* lainnya tanpa adanya agen koordinator. Inilah sari pati dari teknologi swarm. Ditata sedemikian rupa sehingga agen swarm dapat bergerak mandiri dan dapat berinteraksi dengan agen lain di sekitarnya.

Penelitian menggunakan metode *artificial neural network self-organizing map* (ANN-SOM) dilakukan untuk menggerakkan sekumpulan agen *swarm* menuju ke suatu area dan membentuk formasi yang diinginkan. Simulasi dilakukan dengan menggunakan tiga macam formasi yang diinginkan yaitu *spherical, cross* dan *pipe obstruction*. Simulasi dilakukan pada *grid* 4 dan *grid* 5. Simulasi menunjukkan hasil yang baik kecuali pada formasi *pipe obstruction*. Hal ini semakin terlihat pada simulasi *grid* 5. Di sini menunjukkan bahwa algoritma SOM akan memiliki kesulitan ketika formasi yang diinginkan memiliki *hollow spaces* [1].

Orientasi dari frame referensi local pada agen yang tidak selaras antara satu dengan yang lain, dikarenakan tidak adanya *common sense of orientation*. Disusunlah strategi kontrol yang berisikan dua hukum penting yaitu hukum penyearahan orientasi dan hukum pengaturan formasi. Setelah orientasi berhasil disearahkan, maka permasalahan pengaturan formasi akan menjadi mirip dengan permasalahan pengaturan formasi *displacement-based*. Hasil simulasi menunjukkan orientasi berhasil disearahkan dan para agen berhasil membentuk formasi sesuai dengan grafik interaksi yang diinginkan. Sayangnya, grafik interkasi yang dipakai kurang fleksibel sehingga pemilihan posisi dalam formasi akan rigid [2].

Tiap agen menghitung kontrol input menggunakan posisi relatif dari sekumpulan formasi agen sekitar, informasi ini diekspresikan dalam frame koordinat lokal milik agen sendiri tanpa *common reference*. Kontroller berdasar pada meminimalkan fungsi Lyapunov yang ditambah dengan matriks rotasi yang

locally computed, yang mana diperlukan karena tidak adanya *common orientation*. Metode distribusi *coordinate-free* mencapai stabilisasi global pada formasi rigid dengan agen hanya menggunakan sebagian informasi dari tim, tidak memerlukan agen ketua dan dapat diaplikasikan pada agen *single-integrator* atau agen *unicycle*. Dengan menggunakan *local cost-function* yang meminimalkan *cliques* pada sub-formasi, maka akan didapatkan *global cost-function* yang meminimalkan *cliques* pada formasi rigid [3].

Dari beberapa penelitian yang telah penulis kaji, maka penulis mengusulkan penelitian tentang perancangan strategi pembentukan formasi untuk agen menggunakan 2 tingkat algoritma koordinasi agen. Hal ini dengan mengadopsi permasalahan pada pustaka [1] yaitu metode ANN-SOM tidak baik dalam membentuk formasi yang memiliki *hollow-space*. Lalu menggunakan strategi kontrol dari pustaka [2] untuk membentuk formasi dan menggunakan *cost function* dari pustaka [3] untuk meminimalkan jarak yang ditempuh oleh semua agen.

1.2 Rumusan Masalah

Dalam penelitian ini, masalah utama yang dibahas adalah bagaimana agen bergerak dengan perencanaan jalur pada strategi *offline*, lalu bagaimana agen bergerak mengikuti jalur pada strategi *online*, juga bagaimana agen membentuk formasi dan bagaimana agen menempuh akumulasi jarak minimum sampai formasi yang diinginkan terbentuk.

1.3 Tujuan

Tujuan penelitian ini adalah menghasilkan strategi pembentukan formasi yang berhasil membentuk formasi agen sesuai dengan yang diinginkan dan meminimalkan akumulasi jarak tempuh semua agen dalam upaya menghindari halangan.

1.4 Batasan Masalah

Adapun batasan masalah yang akan diteliti adalah sebagai berikut:

1. Agen yang digunakan adalah berupa partikel.
2. Penelitian akan dilakukan dalam bentuk simulasi 2 dimensi.

1.5 Kontribusi

Kontribusi yang penulis berikan dari penelitian ini adalah penambahan algoritma pembentukan formasi dan *cost function* yang meminimalkan akumulasi jarak tempuh agen pada metode ANN-SOM.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

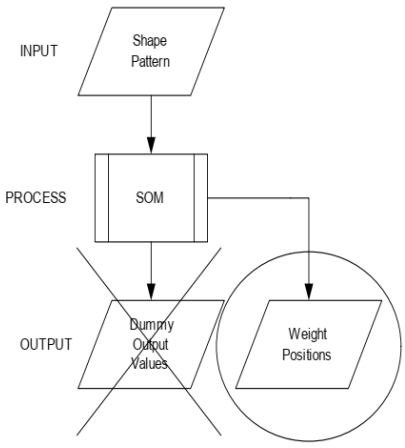
2.1 Kajian Penelitian Terkait

Pada subbab ini diuraikan terkait beberapa penelitian terdahulu yang telah membahas permasalahan yang penulis kerjakan dalam penelitian ini. Sehingga jelas dasar permasalahnnya dan kekurangan yang bisa penulis perbaiki untuk keberhasilan penelitian ini.

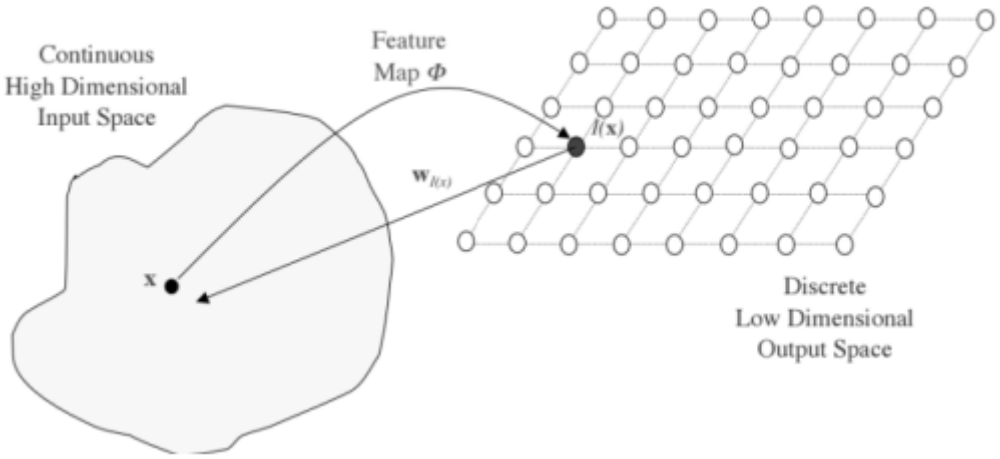
2.1.1 Obstacle Avoidance for Quadrotor Swarm using Artificial Neural Network Self-Organizing Map [1]

Tujuan dari penelitian pada pustaka ini adalah menggunakan ANN-SOM untuk memberikan koordinat yang benar pada tiap *swarm* individual sehingga formasi *swarm* akan didapatkan pada area yang telah ditentukan dengan menghindari halangan yang ada di sekitar.

Sistem yang dipakai digambarkan oleh *Input-Process-Output* (IPO) *Model* seperti pada Gambar 2.1. Sistem menggunakan *dummy output values* sebagai output palsu yang dipakai untuk mendapatkan koreksi bobot. Bobot dari neuron akan menjadi *actual output* yang menjelaskan bagaimana *swarm agents* akan menempatkan dirinya pada lokasi yang telah ditentukan dan pada saat yang sama akan menghindari halangan.



Gambar 2.1 IPO Model dari Sistem



Gambar 2.2 *Input-Output Space Feature Map Mapping*

Feature map dibangun dahulu supaya berfungsi sebagai perantara untuk memetakan *input space* pada *output space* seperti pada Gambar 2.2. Klasifikasi menjadi mungkin ketika *continuous i.s.* berhubungan dengan *discrete o.s.*.

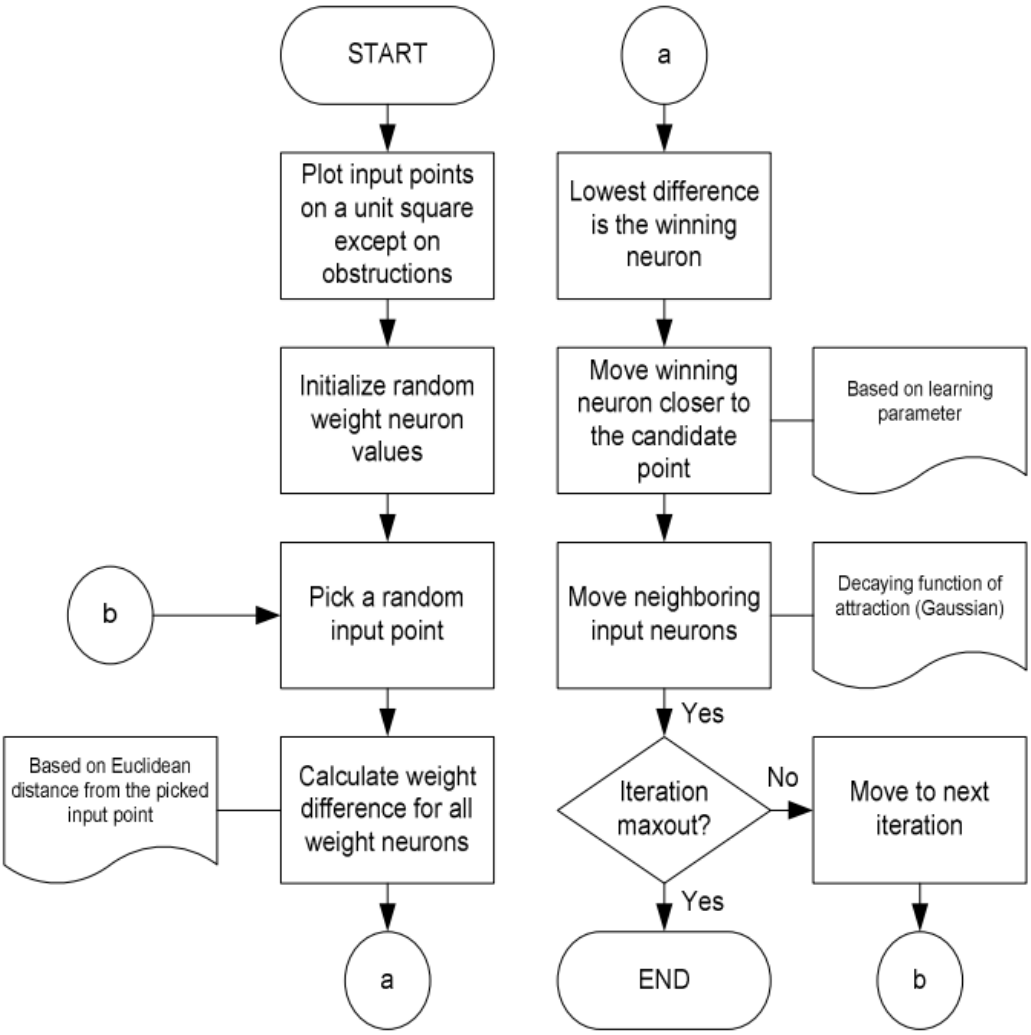
Untuk penelitian ini, bentuk dari *neighbourhood* juga diperhitungkan. Target *neighbourhood* berupa kotak dengan halangan yang berbeda-beda. 3 macam pola akan dipakai dengan bentuk yang unik. Ini semua adalah *input vector.*

Swarming UAV agents diberikan nilai awal acak dengan vektor bobot $\vec{w}$. Vektor acak dalam *input space* akan terpilih dan disebut $\overrightarrow{pick}$. Jarak Euclidean antara $\vec{w}$ dan $\overrightarrow{pick}$ akan dievaluasi untuk tiap neuron dengan menggunakan (2.1)

$$w_{diff} = \left|\vec{w} - \overrightarrow{pick}\right| \tag{2.1}$$

Neuron dengan nilai w_{diff} terkecil maka akan ditetapkan sebagai neuron pemenang. Proses pembelajaran kompetisi ini menjadikan perubahan pemetaan dari *continuous* ke *discrete*. Fungsi neighbourhood dapat diekspresikan sebagai distribusi Gaussian seperti pada (2.2)

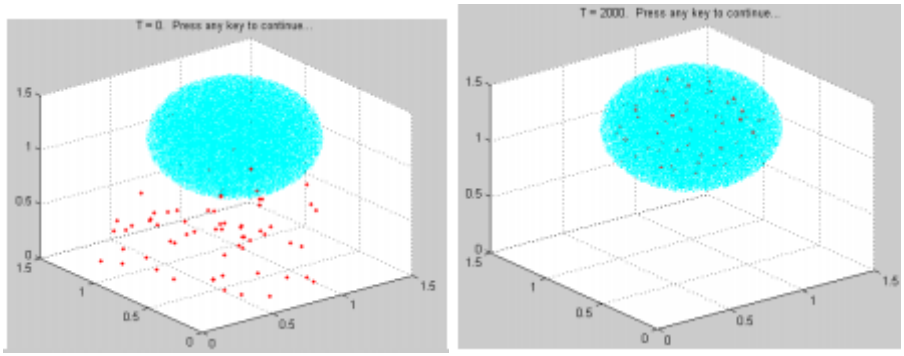
$$gauss = \frac{e^{-\left(\frac{\sum(opick-out)^2}{2*sigma_t^2}\right)}}{sigma_t\sqrt{2*\pi}} \tag{2.2}$$



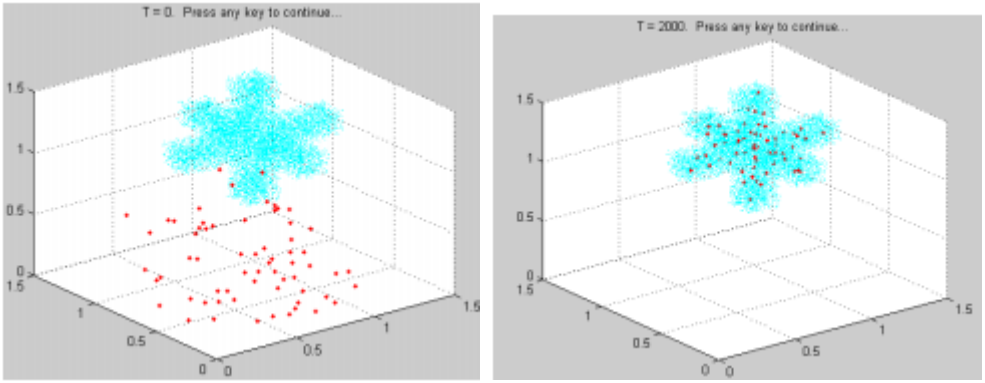
Gambar 2.3 Flowchart dari *Self-Organizing Map*

Selanjutnya adalah proses Adaptif. Proses ini harus diulang beberapa iterasi untuk konvergen pada hasil yang dapat dimengerti. Lalu untuk langkah terakhir dilakukan iterasi ulang secara menyeluruh dari proses kompetitif ke proses adaptif. Algoritma SOM diringkas menjadi flowchart seperti pada Gambar 2.3. Algoritma mulai dengan *plotting input vector*. Kemudian menjalani proses berulang yang mengarah pada estimasi lokal dari bobot neuron (koordinat output dari tiap quadrotor). Algoritma ini akan berhenti ketika banyaknya iterasi telah mencapai batasan maksimum yang telah ditentukan. Hasil *feature map* akan menunjukkan tanda-tanda yang dapat dimengerti.

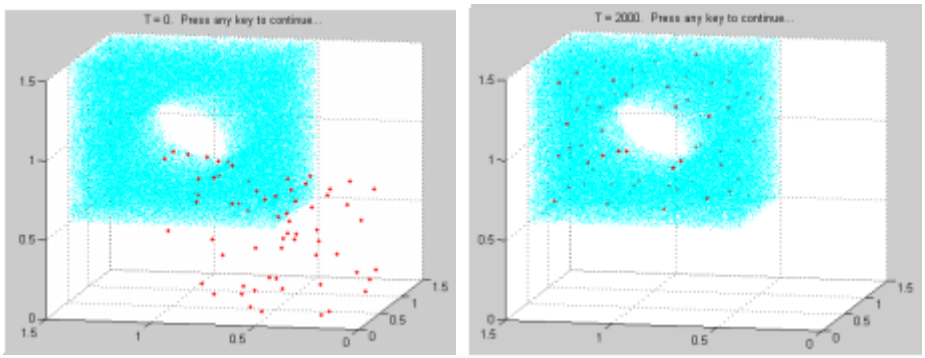
Pengujian dilakukan menggunakan pola bentuk input yang berbeda-beda dan menggunakan ukuran grid yang berbeda. Memakai pola bentuk input *spherical*, *cross* dan *pipe obstruction*. Menggunakan ukuran grid 4 (64 agen quadrotor) dan grid 5 (125 agen quadrotor). Hasil pengujian saat menggunakan grid 4 mendapatkan hasil seperti pada Gambar 2.4, Gambar 2.5 dan Gambar 2.6.



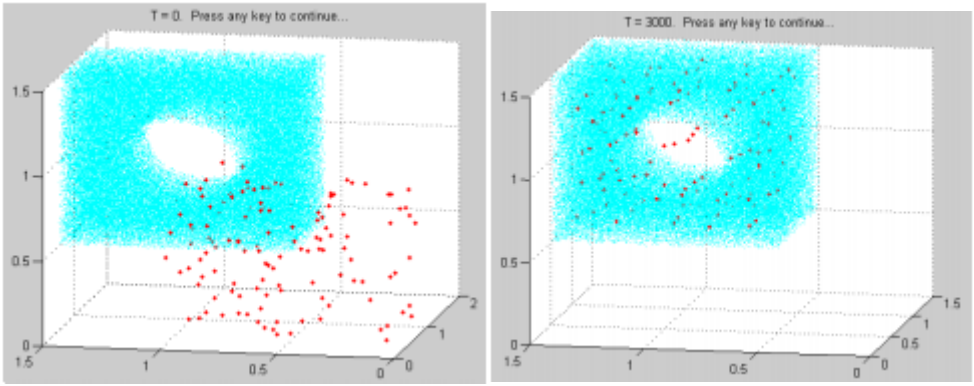
Gambar 2.4 Sebelum-Sesudah Menggunakan Input *Spherical* (Grid 4)



Gambar 2.5 Sebelum-Sesudah Menggunakan Input *Cross* (Grid 4)



Gambar 2.6 Sebelum-Sesudah Menggunakan Input *Pipe Obstruction* (Grid 4)



Gambar 2.7 Sebelum-Sesudah Menggunakan Input *Pipe Obstruction* (Grid 5)

Tetapi saat memakai ukuran grid 5, hasil pengujian pada input *pipe obstruction* memiliki hasil yang kurang baik karena terdapat beberapa agen quadrotor yang menempatkan posisinya di area *obstruction* seperti yang ditunjukkan pada Gambar 2.7.

Kelebihan dari penelitian yang dilakukan pada pustaka ini adalah metode ANN-SOM berhasil mengatur agents (QUAV) untuk membentuk formasi dan menghindari halangan di titik tujuan dengan akurasi yang tinggi. Kekurangan dari penelitian yang dilakukan pada pustaka ini adalah terdapat satu formasi yaitu berupa *pipe obstruction*, metode yang digunakan belum mendapatkan hasil yang memuaskan.

Ide penelitian yang dapat penulis ambil dari pustaka ini adalah menerapkan metode ANN-SOM pada agents untuk mencari titik *centroid* dengan modifikasi menggunakan *Particle Swarm Optimization* yang berfungsi untuk membimbing gerakan agents menuju *centroid*.

2.1.2 Formation Control and Network Localization via Orientation Alignment [2]

Tujuan dari penelitian pada pustaka ini adalah mengajukan strategi pengaturan formasi berdasar pada perpindahan *inter-agent* untuk *single-integrator modeled agents* dalam sebuah plane. Tidak selarasnya orientasi dari frame referensi lokal antar agent, dikarenakan tidak adanya *common sense of orientation*.

Strategi yang diajukan memastikan konvergensi asimtotik formasi agen pada formasi yang diinginkan jika grafik integrasi *uniformly connected* dan semua orientasi awal masih masuk dalam sebuah *interval* dengan *range* kurang dari π terhadap sudut orientasi relatif.

Selain itu, orientasi dari *local reference frame* disearahkan menggunakan strategi yang diajukan. Jika orientasi sudah disearahkan, *common sense of orientation* tersedia pada agent, maka problem pengaturan formasi pada pustaka ini menjadi mirip dengan permasalahan pengaturan formasi *displacement-based*.

Matriks I_n adalah matriks identitas dimensi-n. Dengan N vektor kolom, $X_1 \in \mathbb{R}^{n_1}, \dots\dots, X_N \in \mathbb{R}^{n_N}$, $(X_1, \dots\dots, X_N) \equiv (X_1^T, \dots\dots, X_N^T)^T$. Diberikan dua matriks A dan B, $A \otimes B$ adalah *Kronecker product* dari matriks.

Untuk grup agen, topologi interaksi antar agen dapat dimodelkan oleh grafik berbobot berarah $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, dimana $\mathcal{V}$ dan $\mathcal{E}$ adalah kumpulan *vertices* dan *edges* dan $\mathcal{A}$ adalah peta yang menaruh nomor real positif pada *edges*. Grafik berbobot $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ disebut grafik interaksi grup. Untuk *vertex* $i \in \mathcal{V}$, kumpulan *vertices* yang berdekatan disebut $\mathcal{N}_i$, yang mana disebut juga *neighbors* dari i. Jika $(i, j) \in \mathcal{E}$, maka *vertex* i adalah orang tua dari j dan j adalah anak dari i. Sebuah pohon adalah grafik berarah dimana setiap *vertex* hanya memiliki satu orang tua kecuali untuk satu *vertex*, disebut *the root*, yang tidak memiliki orang tua. Perentangan pohon $\mathcal{G}$ adalah pohon berarah yang berisikan setiap *vertex* dari $\mathcal{G}$. Matriks Laplacian $L = [l_{ij}]$ yang terkait dengan $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ didefinisikan sebagai $l_{ij} = \sum_{k \in \mathcal{N}_i} a_{ik}$ jika $i = j$ dan $l_{ij} = -a_{ij}$ jika tidak.

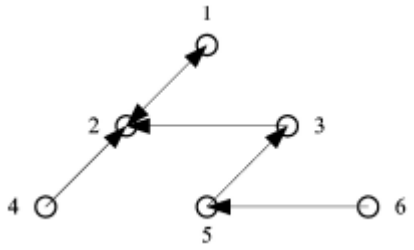
Sebuah grafik berbobot berarah *time-varying* didenotasikan oleh $\mathcal{G}(t) = \big(\mathcal{V}, \mathcal{E}(t), \mathcal{A}(t)\big)$. Diasumsikan $a_{ij}(t) \in \{0\} \cup [a_{min}, a_{max}]$ untuk semua $i, j \in \mathcal{V}$, dimana $0 < a_{min} < a_{max}$ dan a_{max} adalah *finite*, untuk semua $t \geq t_0$. Maka $(i, j) \in \mathcal{E}(t)$ jika dan hanya jika $a_{min} \leq a_{ij}(t) \leq a_{max}$. Grafik $\mathcal{G}(t) = \big(\mathcal{V}, \mathcal{E}(t), \mathcal{A}(t)\big)$ adalah terhubung seragam jika, untuk semua $t \geq t_0$, terdapat sebuah waktu *finite* T dan sebuah *vertex* $i \in \mathcal{V}$ sehingga i adalah akar dari rentangan pohon grafik $\left(\mathcal{V}, U_{\tau \in [t,T]} \mathcal{E}(\tau), \int_t^T \mathcal{A}(t)\, d\tau\right)$.

Dengan mempertimbangkan N-agen yang memiliki grafik interaksi $\mathcal{G}(t) = \left(\mathcal{V}, \mathcal{E}(t), \mathcal{A}(t)\right)$. Sebuah protokol konsensus untuk semua agen sebagai berikut: $\dot{x}_i(t) = \sum_{j \in N_i(t)} a_{ij}(t)\left(x_j(t) - x_i(t)\right)$ untuk semua $i \in \mathcal{V}$, dimana $\dot{x}_i(t) \in \mathbb{R}^n$, sehingga dapat dituliskan dalam bentuk vektor seperti pada (2.3).

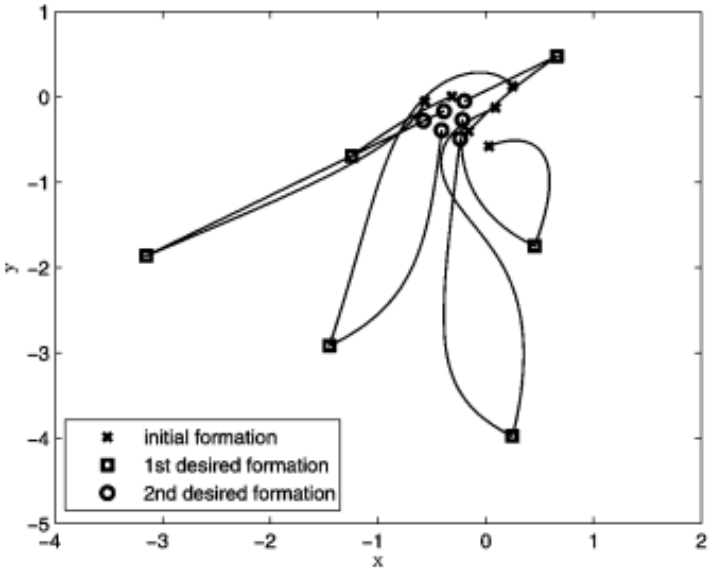
$$\dot{X}(t) = -L(t) \otimes I_n X(t) \tag{2.3}$$

dimana $X(t) = \left(x_1(t), \ldots, x_N(t)\right) \in \mathbb{R}^{nN}$.

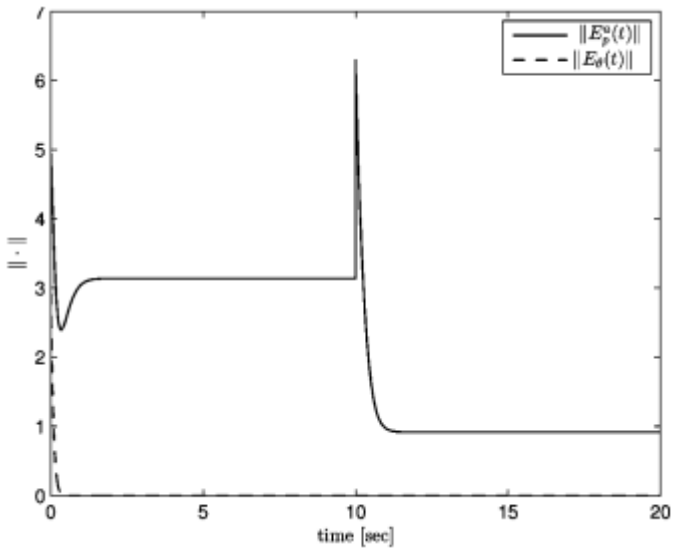
Simulasi dilakukan menggunakan grafik interaksi sesuai dengan Gambar 2.8. Posisi awal agen akan diacak lalu akan bergerak menuju formasi yang diinginkan. Formasi yang diinginkan mengalami satu perubahan.



Gambar 2.8 Grafik Interaksi



Gambar 2.9 Hasil Simulasi Berdasarkan pada Grafik Interaksi



Gambar 2.10 Grafik Kesalahan Formasi $\left\|E_p^a(t)\right\|$ dan Kesalahan *Orientation Alignment* $\left\|E_\theta\ (t)\right\|$

Hasil simulai ditunjukkan oleh Gambar 2.9. Terlihat jelas bahwa setiap agen berhasil bergerak dari formasi awal menuju ke formasi yang diinginkan pertama kemudian lanjut bergerak kembali ke formasi yang diinginkan kedua. Kesalahan *orientation alignment* berhasil konvergen ke zero. Kesalahan *orientation alignment* diukur dengan menggunakan sudut orientasi relative antar agen. Hal ini tidak berubah walaupun saat formasi yang diinginkan dirubah. Hal ini menunjukkan agen yang sudah memiliki *common sense of orientation*, dapat mengatur dirinya untuk membentuk formasi berdasarkan pada *displacement* dirinya.

Kelebihan dari penelitian yang dilakukan pada pustaka ini adalah strategi pengaturan yang diajukan cukup menggunakan pengukuran jarak dan sudut relatif. Strategi pengaturan yang diajukan berhasil memastikan *exponential convergence* dari formasi yang dibentuk. Kekurangan dari penelitian yang dilakukan pada pustaka ini adalah grafik interaksi yang kurang fleksibel sehingga akan menyulitkan jika formasi yang akan dibentuk berupa formasi dinamis.

Ide penelitian yang dapat penulis ambil dari pustaka ini adalah menggunakan pengukuran jarak dan sudut relative untuk membangun strategi pengaturan formasi dengan grafik interaksi yang dimodifikasi.

2.1.3 Distributed Formation Stabilization Using Relative Position Measurements in Local Coordinates [3]

Tujuan dari penelitian pada pustaka ini adalah merancang suatu metode distribusi untuk menstabilkan sekumpulan agen yang bergerak di lingkungan 2 dimensi menuju formasi rigid tertentu. Tiap agent menghitung input kontrol menggunakan posisi relatif dari sekumpulan formasi sekitar yang diekspresikan dalam frame koordinat lokal secara independen tiap agent tanpa membutuhkan referensi yang sama.

Strategi kontrol formasi yang digunakan adalah strategi kontrol yang meminimalkan *global cost function* γ (jumlahan dari fungsi *partial* γ_m yang berhubungan dengan *cliques* maksimal). Tiap agen bergerak di kumpulan *cliques* maksimal miliknya. Di dalam frame koordinasi global *arbitrary*, *cost function* (2.4) dipakai untuk tiap *cliques* maksimal:

$$\gamma_m = \frac{1}{2N_m} \sum_{j \in I_m} \left\| \sum_{k \in I_m} q_{jk} - R_m c_{jk} \right\|^2 \tag{2.4}$$

dengan:

$R_m \in SO(2)$ = matriks rotasi yang bernilai sama untuk tiap agen di *cliques*

c = posisi agen

q = representasi dari c setelah dirotasi sesuai dengan $q = Rc$

M = jumlah maksimal *cliques* dalam G_f

$m = 1, \dots, M$ (m adalah jumlah *cliques*)

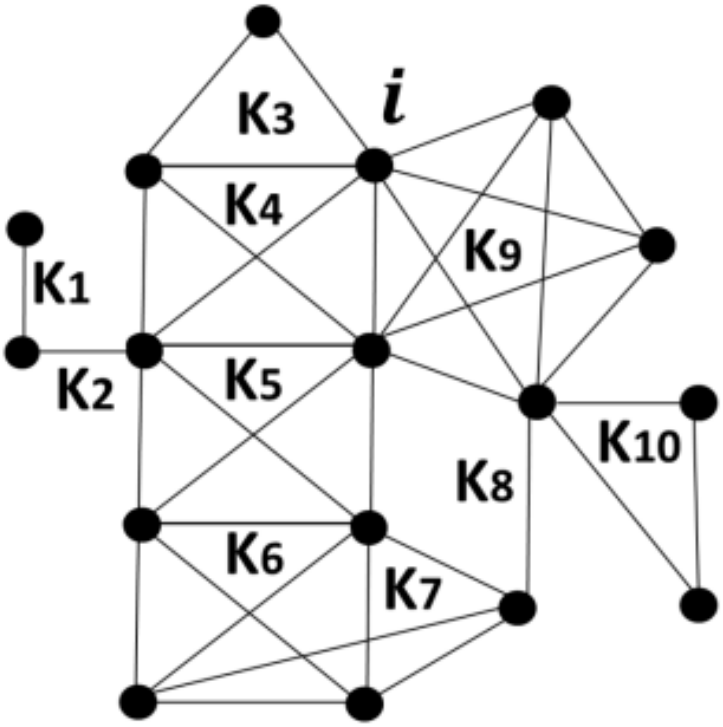
I_m = adalah kumpulan indeks dari titik yang membentuk $m - th\ clique$

Persamaan (2.4) mempunyai *null term* yang terjadi saat $j = k$. Jika $\gamma_m = 0$, maka sesuai dengan (2.3) dua agen dengan $j = i_1$ dan $j = i_2$ memenuhi $\sum_{k \in I_m} q_{i_1 k} - R_m c_{i_1 k}$, $\sum_{k \in I_m} q_{i_2 k} - R_m c_{i_2 k}$. Didapatkan bahwa $q_{i_1 i_2} = R_m c_{i_1 i_2}$ yang berpengaruh untuk setiap pasang i_1, i_2 dalam I_m. Jika $\gamma_m = 0$, maka agen sudah berada di masing-masing sub-formasi yang diinginkan. Terlihat bahwa γ_m

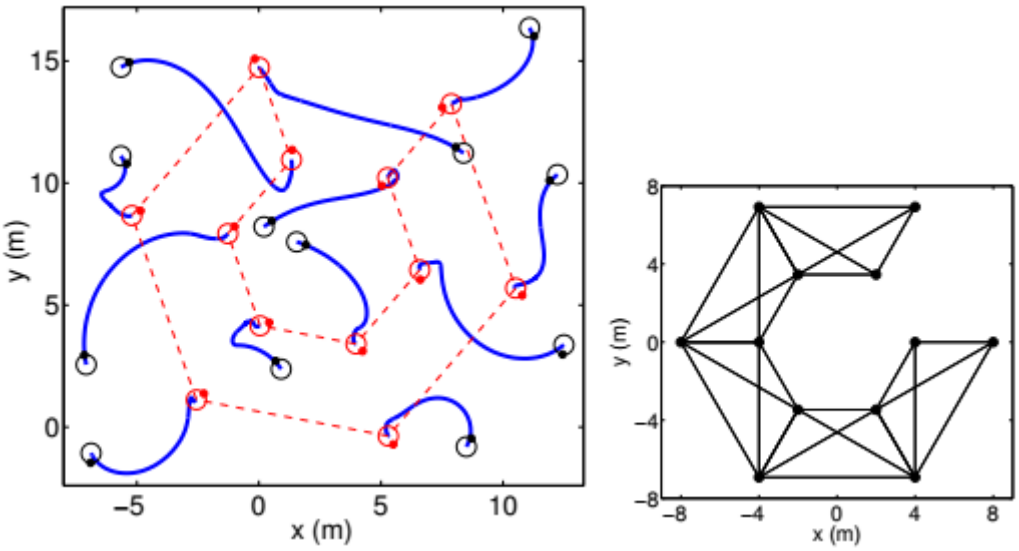
menunjukkan seberapa jauh agen dari sub-formasi yang diinginkan. Untuk *global cost function* menggunakan (2.5)

$$\gamma = \sum_{m=1,\ldots,M} \gamma_m \tag{2.5}$$

Ilustrasi yang menunjukkan struktur dari *cliques* maksimal yang merupakan basis dari kontroller yang diajukan tampil pada Gambar 2.11. Dengan $N = 17$ titik dan $M = 10$ *cliques* maksimal, didenotasikan sebagai $K_m, m = 1, \ldots, M$. Ukuran dari *cliques* maksimal berkisar antara dua sampai lima titik. Contohnya, $K1$ berisikan 2 titik, $K3$ berisikan 3 titik, $K4$ berisikan 4 titik, dan $K9$ berisikan 5 titik. Perhatikan bahwa tiap titik dan tiap ujung dari grafik G_f adalah bagian dari satu atau banyak *cliques* maksimal. Maka, jika $\{i, j\} \in \mathcal{E}$, vektor relatif antara i dan j berkontribusi untuk paling tidak satu γ_m dari (2.4). Ini artinya bahwa dengan melewati semua *cliques* maksimal M, *global cost function* γ meliputi semua ujung dalam grafik G_f.



Gambar 2.11 Grafik Formasi *Arbitrary* G_f



Gambar 2.12 Hasil Simulasi untuk 12 Agen dan Formasi Rigid Tertentu

Simulasi dilakukan menggunakan 12 agen menggunakan formasi rigid tertentu seperti yang ditampilkan oleh Gambar 2.12. Hasil simulasi terletak pada bagian kiri dari Gambar 2.12 dan formasi rigid tertentu terletak pada bagian kanan dari Gambar 2.12.

Kelebihan dari penelitian yang dilakukan pada pustaka ini adalah metode distribusi *coordinate-free* yang diajukan mencapai *global stabilization* pada formasi rigid dengan agen hanya menggunakan sebagian informasi posisi relatif dari keseluruhan team, tidak butuh leader, dan dapat diterapkan pada agen *single-integrator* ataupun *unicycle*. Penggunaan cost-function untuk penentuan posisi adalah kunci utama dalam pustaka ini. Kekurangan dari penelitian yang dilakukan pada pustaka ini adalah grafik formasi masih statis, sehingga strategi control akan banyak dimodifikasi untuk kondisi grafik formasi yang dinamis.

Ide penelitian yang dapat penulis ambil dari pustaka ini adalah menggunakan *cost-function* yang meminimalisasi jarak tempuh agen terhadap penentuan posisi dalam pengaturan formasi.

2.2 Teori Dasar

Pada subbab ini diuraikan terkait beberapa dasar teori yang berkaitan dengan permasalahan yang penulis kerjakan dalam penelitian ini. Sehingga membantu penulis dalam penyusunan metodologi penelitian dan pengujiannya.

2.2.1 Artificial Neural Network [4]

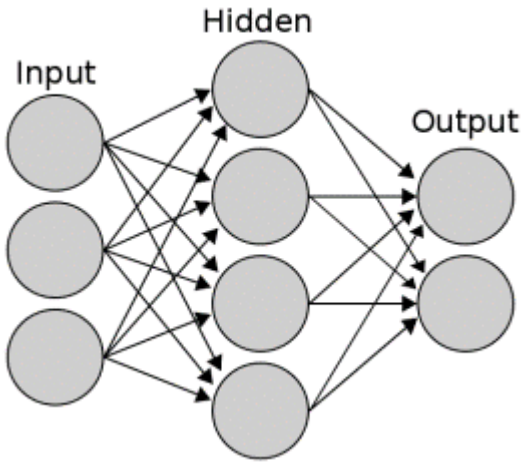
Jaringan saraf tiruan (JST) (Bahasa Inggris: *artificial neural network* (ANN), atau juga disebut *simulated neural network* (SNN), atau umumnya hanya disebut *neural network* (NN), adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan sistem saraf manusia. JST merupakan sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Oleh karena sifatnya yang adaptif, JST juga sering disebut dengan jaringan adaptif.

Secara sederhana, JST adalah sebuah alat pemodelan data statistik nonlinier. JST dapat digunakan untuk memodelkan hubungan yang kompleks antara *input* dan *output* untuk menemukan pola-pola pada data. Menurut suatu teorema yang disebut "teorema penaksiran universal", JST dengan minimal sebuah lapis tersembunyi dengan fungsi aktivasi *nonlinear* dapat memodelkan seluruh fungsi terukur boreal apapun dari suatu dimensi ke dimensi lainnya [7].

Model pada JST pada dasarnya merupakan fungsi model matematika yang mendefinisikan fungsi $f: X \rightarrow Y$. Istilah "jaringan" pada JST merujuk pada interkoneksi dari beberapa neuron yang diletakkan pada lapisan yang berbeda. Secara umum, lapisan pada JST dibagi menjadi tiga bagian:

- Lapis masukan (*input layer*) terdiri dari neuron yang menerima data masukan dari variabel X.
- Lapisan tersembunyi (*hidden layer*) terdiri dari neuron yang menerima data dari lapisan masukan.
- Lapisan luaran (*output layer*) terdiri dari neuron yang menerima data dari lapisan tersembunyi atau langsung dari lapisan masukan yang nilai luarannya melambangkan hasil kalkulasi dari X menjadi nilai Y.

Secara matematis, neuron merupakan sebuah fungsi yang menerima masukan dari lapisan sebelumnya $g_i(x)$ (lapisan ke-i). Fungsi ini pada umumnya mengolah sebuah vektor untuk kemudian diubah ke nilai skalar melalui komposisi nonlinear weighted sum, dimana $f(x) = K(\sum w_i g_i(x))$, K merupakan fungsi khusus yang sering disebut dengan fungsi aktivasi dan w merupakan bobot atau $weight$ [7]. Model ANN ditunjukkan pada Gambar 2.13.



Gambar 2.13 Model Jaringan Syaraf Tiruan

2.2.2 Particle Swarm Optimization [6]

PSO didasarkan pada perilaku sebuah kawanan burung atau ikan. Algoritma PSO meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari tindakan individu dan pengaruh dari individu-individu lain dalam sesuatu kelompok, kata partikel menunjukkan misalnya, seekor burung dalam kawanan burung. setiap individu atau partikel berperilaku dengan cara menggunakan kecerdasannya sendiri dan juga dipengaruhi perilaku kelompoknya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh dari kelompok tersebut.

Dalam *particle swarm optimization* (PSO), kawan diasumsikan mempunyai ukuran tertentu dengan posisi awal setiap partikel bersifat acak dalam

ruang multidimensi. Setiap partikel diasumsikan dua karakteristik yaitu, posisi dan kecepatan. Setiap partikel dalam ruang bergerak dalam posisi tertentu dan mengingat posisi terbaik yang dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi tersebut.

Meskipun setiap burung memiliki keterbatasan dalam hal kecerdasan, biasanya ia akan mengikuti kebiasaan seperti berikut:

1. Seekor burung tidak akan terlalu dekat dengan burung yang lain.
2. Burung tersebut akan mengarahkan terbangnya ke arah rata-rata keseluruhan burung.
3. Akan memposisikan diri dengan rata-rata posisi burung yang lain dengan menjaga jarak antar burung dalam kawanan itu sehingga tidak terlalu jauh.

Dengan demikian perilaku kawanan burung didasarkan dari kombinasi tiga faktor sebagai berikut:

1. Kohesi - terbang bersama.
2. Separasi – jangan terlalu dekat.
3. Penyesuaian (*alignment*) – mengikuti arah bersama.

Jadi PSO dikembangkan dengan berdasarkan pada model berikut:

1. Kriteria seekor burung mendekati target atau makanan (atau bisa minimum atau maksimum suat fungsi tujuan) secara cepat mengirim informasi kepada burung-burung yang lain dalam kawasan tertentu.
2. Burung yang lain akan mengikuti arah menuju ke makan tetapi tidak secara langsung.
3. Ada komponen yang tergantung pada pikiran setiap burung, yaitu memorinya tentang apa yang sudah dilewat sebelumnya.

Pada algoritma PSO ini, pencarian solusi dilakukan oleh suat populasi yang terdiri dari beberapa partikel. Populasi dibangkitkan secara *random* dengan batasan permasalahan yang dihadapi. Setiap partikel mempresentasikan partikel atau solusi dari permasalahan yang dihadapi. Setiap partikel melakukan pencarian solusi yang optimal dengan melintasi ruang pencarian. Hal ini dilakukan dengan cara setiap partikel melakukan penyesuain terhadap posisi terbaik dari setiap

partikel tersebut (*local best*) dan posisi partikel terbaik dari seluruh kawanan (*global best*) selama melintasi ruang pencarian. Jadi penyebaran pengalaman atau informasi terjadi dalam partikel itu sendiri dan antara suatu partikel dengan partikel terbaik dari seluruh kawanan selama proses pencarian solusi. Setelah itu, dilakukan proses pencarian untuk mencari posisi terbaik setiap partikel dalam jumlah iterasi tertentu sampai didapatkan posisi relatif yang *steady* atau mencapai batas iterasi yang telah ditetapkan. Pada setiap iterasi, setiap solusi yang direpresentasikan oleh posisi partikel, dievaluasi performanya dengan cara memasukan solusi tersebut ke dalam *fitness function*.

Setiap partikel diperlakukan seperti titik pada suatu dimensi ruang tertentu kemudian terdapat dua faktor yang memberikan karakter terhadap status partikel pada ruang pencarian yaitu posisi partikel dan kecepatan partikel. Persamaan (2.6) dan (2.7) merupakan formulasi matematika yang menggambarkan posisi dan kecepatan partikel suatu dimensi ruang tertentu,

$$X_i(t) = x_{i'0}(t), x_{i1}(t), ..., x_{iN}(t) \tag{2.6}$$

$$V_i(t) = v_{i'0}(t), v_{i1}(t), ..., v_{iN}(t) \tag{2.7}$$

dengan:

X = posisi partikel
V = kecepatan partikel
i = indeks partikel
t = iterasi ke-t
N = ukuran dimensi ruang

Persamaan (2.8) dan (2.9) merupakan model matematika yang menggambarkan mekanisme updating status partikel Kennedy dan Eberhert:

$$V_i(t) = v_{i'}(t-1) + c_1 r_1 (X_i L - X_i(t-1)) + c_2 r_2 (X_i G - X_i(t-1)) \tag{2.8}$$

$$X_i(t) = V_{i'}(t) + X_i(t-1) \tag{2.9}$$

dengan:

X_i = local best dari partikel ke-i

X_iG = *global best* dari seluruh kawanan

$c_1\ dan\ c_2$ = *learning factor*

r_1 = bilangan random yang bernilai antara 0 sampai 1

r_2 = bilangan random yang bernilai antara 0 sampai 1

Persamaan (2.8) digunakan untuk menghitung kecepatan partikel yang baru berdasarkan kecepatan sebelumnya, jarak antara posisi saat ini dengan posisi terbaik partikel (*local best*), dan jarak antara posisi saat ini dengan posisi terbaik kawanan (*global best*). Kemudian partikel terbang menuju posisi yang baru berdasarkan (2.9). Setelah algoritma PSO ini dijalankan dengan sejumlah iterasi tertentu hingga mencapai kriteria pemberhentian, maka akan didapatkan solusi yang terletak pada *global best*.

Algoritma PSO meliputi langkah berikut:

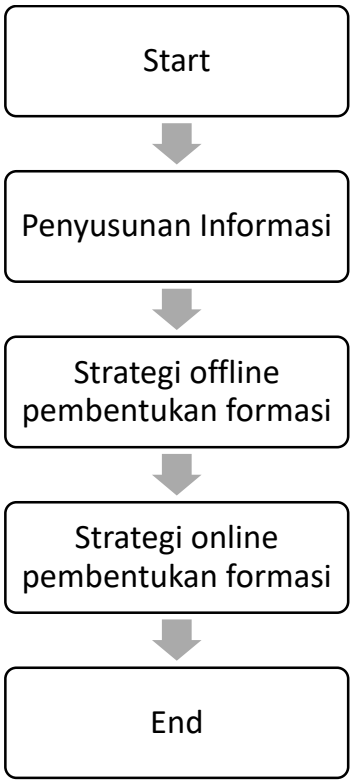
1. Membangkitkan posisi awal sejumlah partikel sekaligus kecepatan awalnya secara *random*.
2. Mengevaluasi *fitness* dari masing-masing partikel berdasarkan posisinya.
3. Menentukan partikel dengan *fitness* terbaik dan tetapkan sebagai G_{best}. Untuk setiap partikel G_{best} awal sama dengan posisi awal.
4. Mengulangi langkah berikut sampai pemberhentian kriteria dipenuhi.
5. Menggunakan P_{best} dan G_{best} yang ada, perbarui kecepatan setiap partikel menggunakan (2.8). Lalu dengan kecepatan baru yang didapat, perbarui posisi setiap partikel menggunakan (2.9).
6. Mengevaluasi *fitness* setiap partikel.
7. Menentukan partikel dengan *fitness* terbaik, dan tetapkan sebagai G_{best}. Untuk setiap partikel tentukan P_{best} dengan membandingkan posisi sekarang dengan P_{best} dari iterasi sebelumnya.
8. Mengecek pemberhentian kriteria bila dipenuhi berhenti, bila sebaliknya kembali ke langkah 1.

BAB 3
METODE PENELITIAN

Pada bab ini diuraikan tahapan-tahapan yang dilakukan dalam proses perancangan sistem. Proses perancangan sistem meliputi perancangan metode pergerakan agen dan strategi pembentukan formasi.

3.1 Perancangan Metode Pergerakan Agen dan Strategi Pembentukan Formasi

Gambar 3.1 menunjukkan tahapan dalam perancangan metode pergerakan agen dan strategi pembentukan formasi. Bentuk formasi dan posisi masing-masing agen sudah diketahui. Posisi awal tiap agen diatur sesuai dengan keadaan tertentu. Strategi pembentukan formasi dibagi menjadi dua tahap yaitu tahap *offline* dan tahap *online*. Terdapat algoritma menghindari tabrakan yang tertanam pada tiap agen.



Gambar 3.1 Garis Besar Tahapan Perancangan Sistem

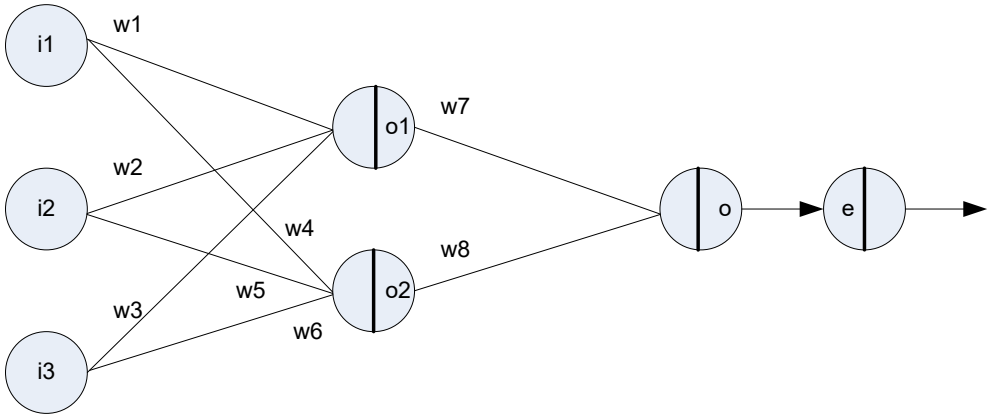
Proses pembentukan formasi akan terus berjalan sampai tiap agen berhasil membentuk formasi. Tiap agen akan menempuh jarak total yang minimal ketika berhasil mencapai formasi yang diinginkan.

3.2 Penyusunan Informasi Posisi Awal, Posisi Tujuan dan Kecepatan Gerak Agen

Posisi awal tiap agen diatur sesuai dengan keadaan tertentu. Hal ini dimaksudkan untuk memudahkan proses analisa pada sesi selanjutnya. Formasi yang diinginkan sudah ditentukan sebelumnya. Setiap formasi yang sudah dipilih pasti memiliki koordinat bagi tiap posisi dalam formasinya. Koordinat ini nantinya akan dipakai sebagai posisi tujuan bagi tiap agen.

Untuk gerak agen akan ditentukan oleh ANN-SOM [1]. Nilai awal dari bobot pada *input-hidden neuron* adalah posisi awal tiap agen yaitu bobot $w_1, w_2, \dots, w_6$ dan menggunakan bias b_1, b_2, b_3. Neural Network akan menghitung koreksi untuk tiap bobot sampai mendapatkan kesalahan yang paling kecil. Sehingga dapat dirumuskan besarnya kecepatan untuk gerak tiap agen [4].

Perancangan sistem koreksi gerak agen menggunakan Neural Network – Algoritma *Back Propagation*. Algoritma akan mengkoreksi bobot pada tiap *neuron* sampai seluruh *epoch* tercapai. Memakai 3 *input*, 2 *hidden neuron* dan 1 *output* dalam penyusunan algoritma. Hasil revisi bobot setelah selesai seluruh *epoch* digunakan untuk koreksi gerak agen. Skema algoritma *back propagation* disusun sesuai dengan Gambar 3.2.



Gambar 3.2 Skema Penyusunan Algoritma *Back Propagation*

3.3 Perancangan Strategi Offline

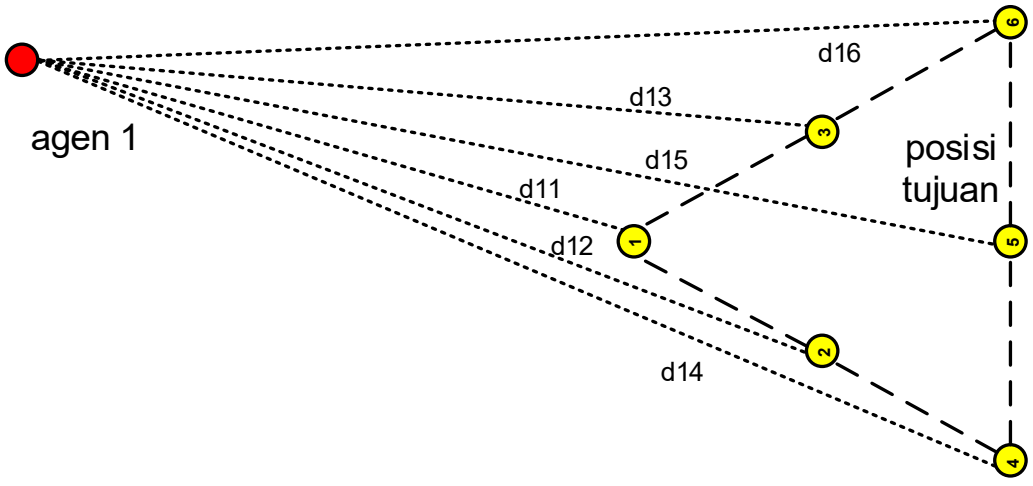
Strategi offline disusun untuk memudahkan agen memilih posisi tujuan dalam formasi yang diinginkan. Dalam strategi offline ini, terdapat dua tahapan yaitu tahap inisialisasi dan kompetisi. Berikut penjelasan untuk masing-masing tahapan tersebut:

- Inisialisasi

Tahapan ini bertujuan untuk menghasilkan gerakan untuk tiap agen pada saat strategi online dijalankan. Gerakan awal dari tiap agen dihasilkan dengan menjalanan ANN-SOM. Posisi awal dari agen akan dipelajari oleh ANN-SOM sampai bergerak pada posisi selanjutnya. Dari semua agen, gerakan agen yang terbaiklah yang akan diambil sebagai gerakan tiap agen untuk dipakai dalam strategi online.

- Kompetisi

Dalam pemilihan posisi tujuan pada tiap agen menggunakan sistem kompetisi. Sistem kompetisi ini dijalankan dengan kriteria akumulasi jarak tempuh seluruh agen minimum. Tiap agen dihitung berapa jarak yang ditempuh untuk menuju ke semua posisi tujuan. Jarak tiap agen dari posisi awal ke posisi tujuan diberi notasi $d_{i1}, d_{i2}, \dots, d_{i6}$. Jadi kalau terdapat agen 1, maka jaraknya terhadap semua posisi tujuan adalah $d_{11}, d_{12}, \dots, d_{16}$. Untuk lebih jelasnya dapat dilihat pada Gambar 3.3.



Gambar 3.3 Perhitungan Jarak tiap Agen terhadap Semua Posisi Tujuan

Informasi jarak tempuh semua agen ini akan dibandingkan untuk mendapatkan akumulasi jarak tempuh minimum. Ketika sudah didapatkan pasangan posisi awal dan posisi tujuan dari setiap agen, maka kompetisi akan berakhir. Kompetisi menggunakan algoritma permutasi yaitu menghitung semua kemungkinan kombinasi urutan tujuan akhir tanpa adanya kombinasi urutan yang sama. Perhitungan permutasi menggunakan (3.1) di bawah ini.

$$P(n,k) = \frac{n!}{(n-k)!} \tag{3.1}$$

dengan:

$P(n,k)$ = banyaknya kombinasi hasil dari permutasi
n = banyaknya data yang tersedia untuk diolah
k = banyaknya data yang dipakai untuk permutasi

Terdapat 6 posisi tujuan (n = 6) dan keenamnya (k = 6) dicari semua kemungkinan kombinasi untuk dipasangkan dengan tiap agen. Maka didapatkan hasil perhitungan permutasi adalah 6! atau 720 kombinasi. Dari 720 kombinasi ini akan dihitung akumulasi jarak tempuh semua agen. Kombinasi yang memiliki akumulasi jarak tempuh paling kecil (minimum) akan dipakai sebagai pilihan posisi tujuan bagi tiap agen.

3.4 Perancangan Strategi Online

Strategi online disusun untuk menggerakkan agen dari posisi awal menuju posisi tujuan sesuai dengan hasil dari strategi offline. Dalam strategi online ini, terdapat dua tahapan yaitu tahap gerak agen dan pemenuhan kriteria. Berikut penjelasan untuk masing-masing tahapan tersebut:

- Gerak Agen

Tiap agen akan bergerak sesuai dengan informasi gerakan awal dari ANN-SOM pada strategi offline. Dalam penyusunan algoritma, posisi agen

direpresentasikan sebagai $p_{1,\ldots,n}(x_{1,\ldots,n}, y_{1,\ldots,n})$. Pergerakan dari satu titik ke titik selanjutnya menggunakan *Euclidean Distance* sesuai dengan (3.2).

$$d_{kl} = \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2} \tag{3.2}$$

Titik-titik yang harus dilalui sudah ditentukan dan dihitung pada strategi offline. Sehingga tiap agen cukup bergerak sesuai dengan titik-titik yang telah ditentukan dengan kecepatan yang konstan untuk mencapai posisi tujuan.

- Memenuhi Kriteria Jarak

Jarak yang ditempuh tiap agen dicatat oleh sistem. Saat agen sudah mencapai posisi tujuan, maka system akan mencatat akumulasi jarak tempuh semua agen dan memastikan apakah sudah memenuhi kriteria minimum. Untuk memenuhi kriteria akumulasi jarak tempuh agen minimum, maka harus memenuhi *cost function* C sesuai dengan (3.3) dan (3.4). *Cost function* C_i adalah *cost function* pada masing-masing agen.

$$C_i = \sum_{j=1}^{m} (p_i(j+1) - p_i(j)) \tag{3.3}$$

$$C = \sum_{i=1}^{6} C_i \tag{3.4}$$

Untuk menghindari terjadinya tabrakan antar agen saat melintasi jalur yang berdekatan maka dirancanglah algoritma pembelokan arah saat jarak antar agen berada di bawah batas *treshold*. Algoritmanya sebagai berikut:

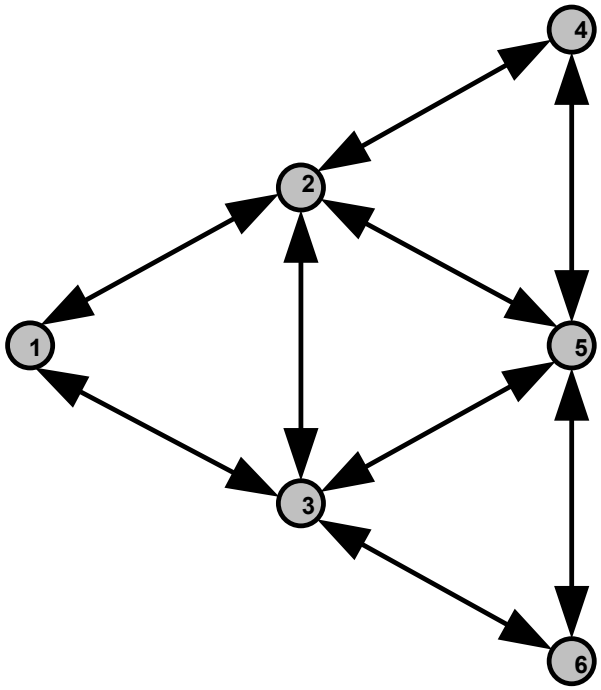
1. Posisi awal agen telah ditentukan.
2. Agen bergerak menuju tujuan masing-masing.
3. Hitung nilai jarak antar agen.
4. Dengan batas *treshold* adalah 0.2, jika terdapat agen yang berdekatan dengan jarak diantaranya di bawah batas *treshold* maka agen ke-*n-1* akan melakukan pembelokan.

5. Pembelokan dilakukan dengan menambahkan posisi x-*axis* dan y-*axis* menggunakan $[0.3 \quad -0.2]$. Kolom pertama untuk pembelokan x-*axis* dan kolom kedua untuk pembelokan y-*axis*.
6. Proses selanjutnya akan berulang pada langkah ke-2.
7. Algoritma akan terus berjalan hingga semua agen sampai di posisi tujuan atau ketika mencapai iterasi maksimal.

3.5 Perancangan Grafik Interaksi antar Agen

Terdapat interaksi antar agen untuk bergerak menuju posisi akhir pada formasi yang diinginkan. Interaksi antar agen ini diatur dalam grafik interaksi. Grafik interaksi yang digunakan berdasarkan pada formasi segitiga dan menggunakan 6 agen seperti yang ditampilkan pada Gambar 3.4.

Grafik interaksi disusun berdasarkan pada besarnya sudut antar agen terhadap sudut referensi relatif. Hal ini berguna untuk menata tiap agen supaya menempati posisinya dengan benar dan menjaga agen supaya tidak masuh ke dalam batas formasi.



Gambar 3.4 Grafik Interaksi antar Agen

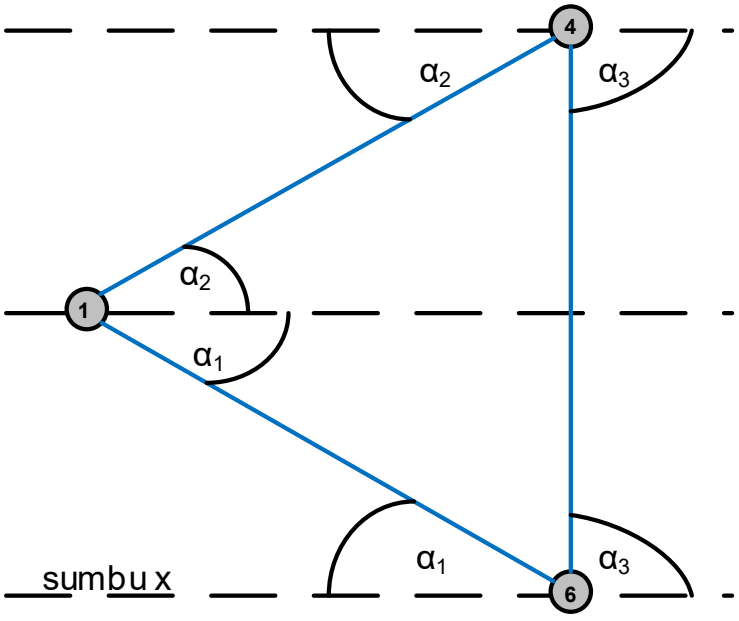
Jarak antar agen dihitung menggunakan (3.2) sehingga didapatkan jarak yang dilambangkan dengan $d_{16}, d_{14}, d_{46}, d_{23}, d_{25}, d_{53}$. d_{16} adalah jarak antar agen 1 dan 6 dan begitu seterusnya sampai d_{53}. Besarnya sudut dapat diukur dengan menggunakan rumus invers tangen (arctan) seperti ditunjukkan pada (3.5).

$$\alpha_i = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \times \frac{180^\circ}{\pi} \tag{3.5}$$

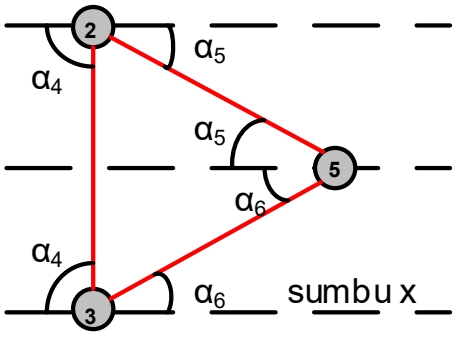
dengan:

α_i = besarnya sudut antara garis d_{kl} dan sumbu x (dalam derajat)
y_2 = posisi koordinat agen kedua pada sumbu y
y_1 = posisi koordinat agen pertama pada sumbu y
x_2 = posisi koordinat agen kedua pada sumbu x
x_1 = posisi koordinat agen pertama pada sumbu x

Sudut yang terbentuk pada formasi agen ditampilkan pada Gambar 3.5 dan 3.6. Terdapat 6 buah sudut yang terbentuk antara garis jarak antar agen d_{kl} dengan sumbu x yaitu $\alpha_1, \alpha_2, \ldots, \alpha_6$.



Gambar 3.5 Sudut-sudut dalam Formasi Agen 1, 4 dan 6

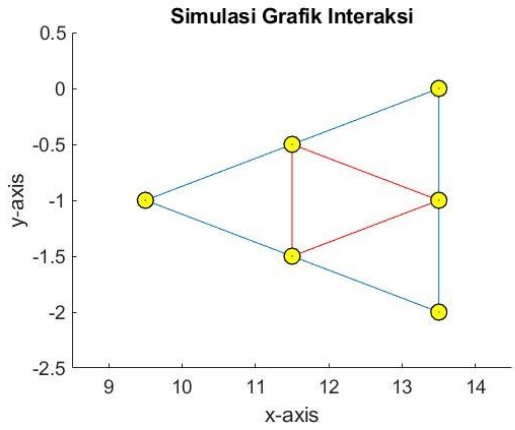


Gambar 3.6 Sudut-sudut dalam Formasi Agen 2, 3 dan 5

Berdasarkan Gambar 3.5 dan 3.6, besarnya sudut yang tercipta pada tiap agen dapat dihitung. Agen 1 memiliki sudut $\alpha_2 + \alpha_1$, agen 4 memiliki sudut $\alpha_3 - \alpha_2$, agen 6 memiliki sudut $\alpha_3 - \alpha_1$, agen 2 memiliki sudut $\alpha_4 - \alpha_5$, agen 3 memiliki sudut $\alpha_4 - \alpha_6$, agen 5 memiliki sudut $\alpha_4 + \alpha_6$.

Pada saat agen bergerak menuju posisi tujuannya masing-masing, algoritma akan terus menghitung besarnya sudut tiap agen. Jika sudut tiap agen sudah sesuai dengan yang diinginkan maka algoritma akan berhenti dan semua agen dipastikan sudah menempati posisinya di formasi yang diinginkan. Jika sudut tiap agen masih belum mencapai kriteria, maka algoritma akan lanjut berjalan sampai sudut yang diinginkan tercapai.

Besarnya sudut yang diinginkan tiap agen untuk formasi segitiga adalah 28.0725° (agen 1), 75.9638° (agen 2), 75.9638° (agen 3), 75.9638° (agen 4), 28.0725° (agen 5) dan 75.9638° (agen 6) sesuai dengan Gambar 3.7.



Gambar 3.7 Simulasi Grafik Interaksi

3.6 Berakhirnya Algoritma

Algoritma akan berakhir jika kondisi tertentu sudah terpenuhi. Kondisi tersebut diantaranya:

- Tiap agen sudah mencapai posisi tujuan.
- Sudut yang terbentuk antar agen sudah memenuhi persyaratan.
- Batas maksimal iterasi sudah terpenuhi.

Semua kondisi itu harus terpenuhi. Jika terdapat satu saja kondisi yang tidak terpenuhi maka sistem akan meneruskan algoritma.

Keseluruhan algoritma ANN-SOM dengan strategi 2 tingkat ditunjukkan pada Tabel 3.1.

Tabel 3.1 Algoritma ANN-SOM dengan Strategi 2 Tingkat

1. Menentukan posisi awal p_i dan posisi tujuan p_f sesuai dengan formasi

   $$p_i = \begin{bmatrix} xi_1 & xi_2 & xi_3 & xi_4 & xi_5 & xi_6 \\ yi_1 & yi_2 & yi_3 & yi_4 & yi_5 & yi_6 \end{bmatrix}$$

   $$p_f = \begin{bmatrix} xf_1 & xf_2 & xf_3 & xf_4 & xf_5 & xf_6 \\ yf_1 & yf_2 & yf_3 & yf_4 & yf_5 & yf_6 \end{bmatrix}$$

2. Inisialisasi bobot $w_1, w_2, \dots, w_6$ dan bias b_1, b_2, b_3
3. Hitung nilai eksitasi net_1, net_2 dan net_3 dengan unit output masing-masing neuron o_1, o_2 dan o_3

   $$net_i = \sum x_i w_i + b$$

   menggunakan fungsi aktivasi *sigmoid*

   $$o_i = f(net_i) = \frac{1}{1 + \exp(-net_i)}$$

4. Hitung feedforward sampai menemui error e terhadap target t

   $$e = o - t$$

5. Hitung error backpropagation δ

   $$\delta_j = o_j(1 - o_j)(o_j - tg)$$

6. Hitung besarnya revisi bobot Δw, dengan γ adalah *learning rate*

   $$\Delta w = -\gamma o_i \delta_j$$

Tabel 3.2 Algoritma ANN-SOM dengan Strategi 2 Tingkat - lanjutan

7. Perbaharui bobot pada masing-masing layer

   $w_{k+1} = w_i + \Delta w$

8. Kembali hitung feedforward dan seterusnya sampai iterasi tertentu
9. Mengambil hasil revisi bobot untuk dijadikan koreksi gerak agen NN_v

   $NN_v = |w_i - w_{koreksi}|$

10. Hitung perubahan posisi agen berdasarkan pada kecepatan gerak agen

    $p_{new} = (p_f - p_i) \times T \times NN_v + p_{old}$

11. Menghitung jarak dari tiap agen d ke semua titik posisi tujuan

    $d_{kl} = \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2}$

12. Menghitung semua kemungkinan jarak tiap agen menggunakan permutasi

    $out = perms([1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6])$

    $pf_i = [pf(out(1)) \quad pf(out(2)) \quad pf(out(3)) \quad pf(out(4)) \quad pf(out(5)) \quad pf(out(6))]$

13. Memilih titik tujuan dengan akumulasi jarak yang minimal

    $pf_{new} = \min(pf_i)$

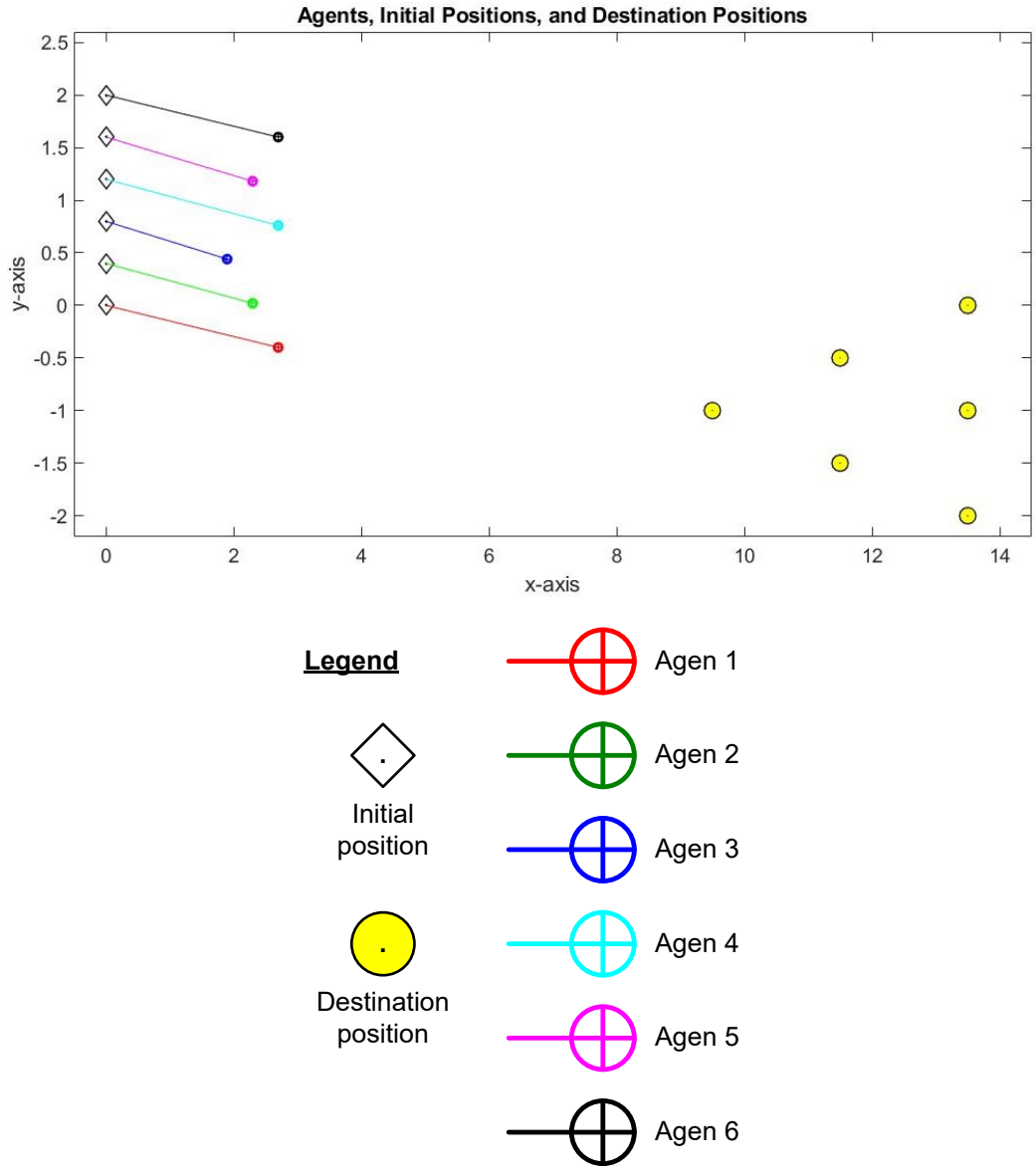
14. Menjalankan agen sesuai dengan kecepatan dan posisi akhir yang sudah ditentukan
15. Menghitung jarak antar agen untuk menghindari terjadinya tabrakan
16. Batas *treshold* adalah 0.2. Jika jarak antar agen ≤ 0.2, maka akan melakukan pembelokan sebesar $[0.3 \quad -0.2]$
17. Menghitung besarnya sudut pada tiap agen

    $$\alpha_i = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \times \frac{180^\circ}{\pi}$$

18. Agen berjalan sampai mencapai posisi tujuan dan besarnya sudut tiap agen sudah tercukupi atau sampai iterasi maksimal tercapai

BAB 4
HASIL DAN PEMBAHASAN

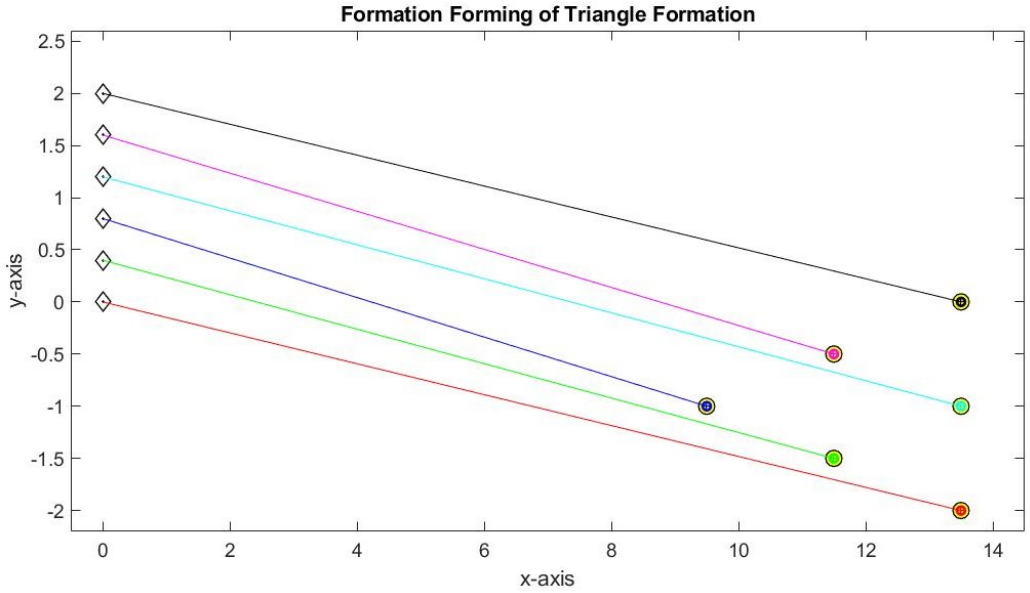
Pengujian dilakukan dengan menggunakan simulasi pada MATLAB 2018a. Dalam penelitian ini, simulasi dilakukan pada bidang 2 dimensi. Agen yang dipakai dalam penelitian ini adalah berupa partikel. Jumlah agen adalah 6 unit. Posisi awal dari tiap agen telah ditentukan dengan posisi tujuan tiap agen diambil dari koordinat posisi dari formasi tujuan seperti yang ditunjukkan pada Gambar 4.1.



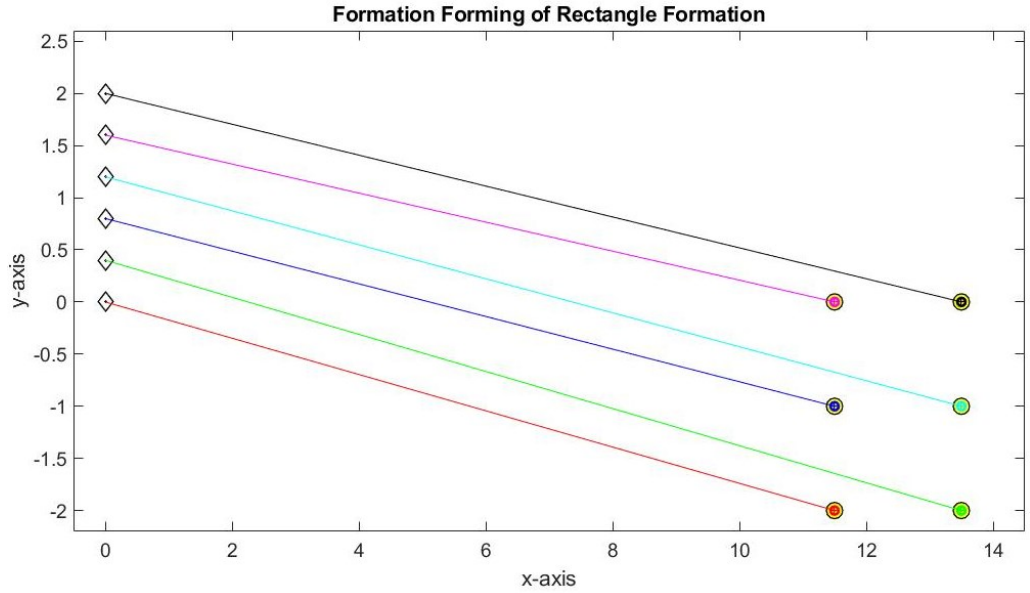
Gambar 4.1 Inisialisasi untuk Simulasi dan Legend

4.1 Pembentukan Formasi

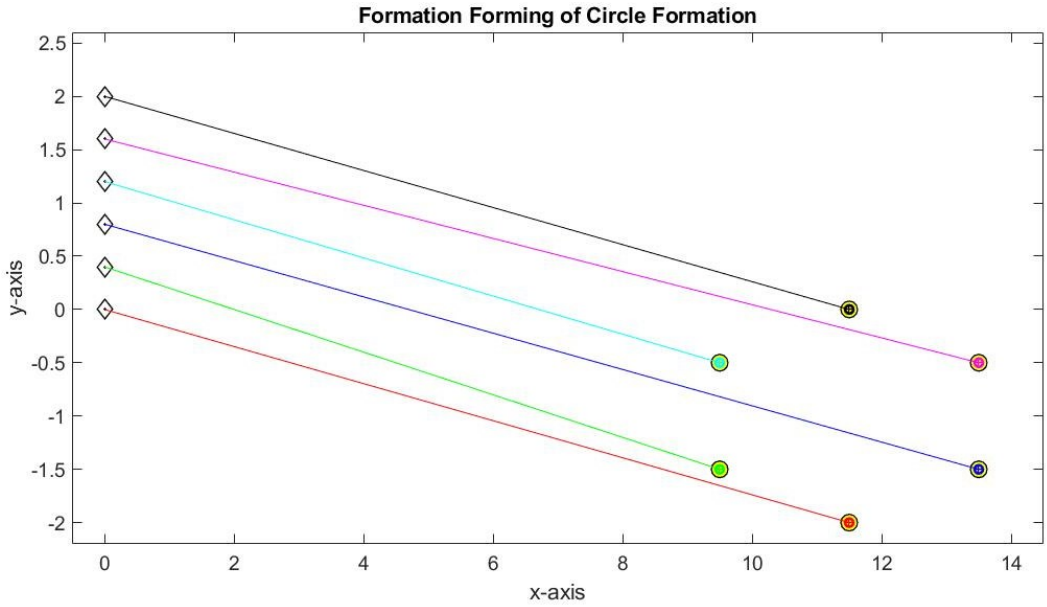
Untuk pengujian bagian pertama adalah uji kemampuan algoritma untuk mengatur agen untuk membentuk formasi sesuai dengan yang diinginkan. Dicoba tiga jenis formasi yaitu formasi segitiga, formasi persegi panjang dan formasi lingkaran. Hasil simulasi dapat dilihat pada Gambar 4.2-4.



Gambar 4.2 Pembentukan Formasi pada Formasi Segitiga



Gambar 4.3 Pembentukan Formasi pada Formasi Persegi Panjang



Gambar 4.4 Pembentukan Formasi pada Formasi Lingkaran

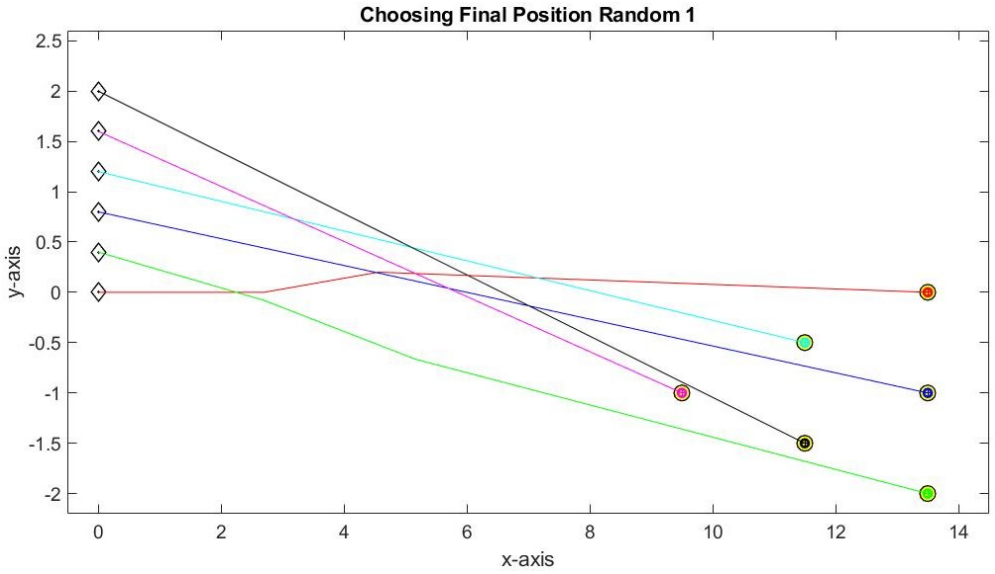
Dari ketiga pengujian di atas terlihat bahwa tiap agen berhasil bergerak menuju posisinya masing-masing sesuai dengan posisi tujuan pada formasi yang diinginkan.

4.2 Pemilihan Posisi Sesuai Kriteria

Pengujian tahap kedua dilakukan untuk menguji kemampuan algoritma untuk mengatur agen memilih posisi tujuannya untuk mendapatkan akumulasi jarak tempuh semua agen adalah minimum. Pengujian dilakukan dengan membandingkan 3 keadaan yaitu:

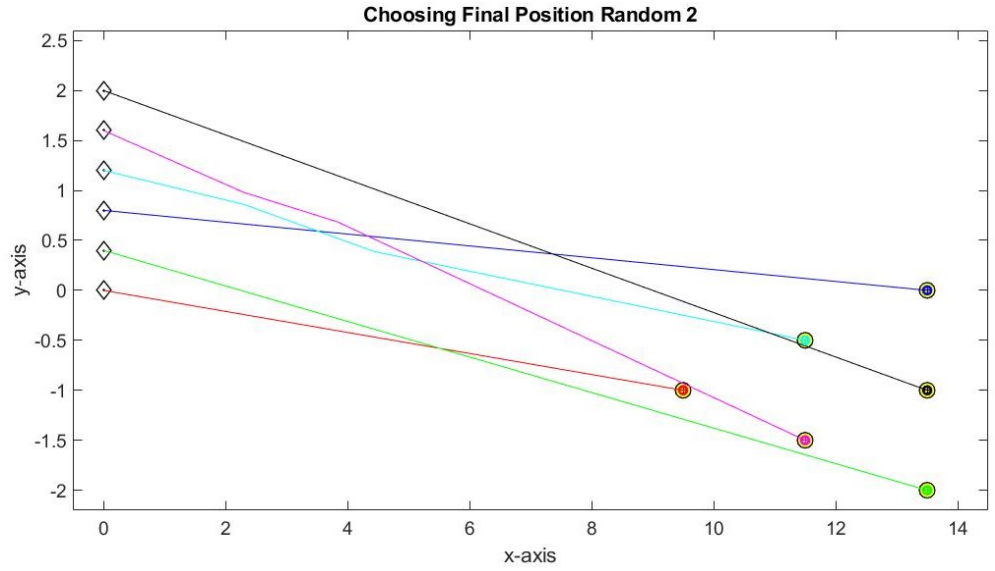
- Keadaan pertama yaitu pengujian dengan pemilihan tujuan akhir adalah acak.
- Keadaan kedua yaitu pengujian dengan pemilihan tujuan akhir sesuai dengan agen yang terdekat.
- Keadaan ketiga yaitu pengujian dengan algoritma yang diusulkan.

Hasil pengujian untuk keadaan pertama ditunjukkan pada Gambar 4.5-8.



Gambar 4.5 Pemilihan Tujuan Akhir Acak 1

Dari Gambar 4.5, jarak tempuh dari masing-masing agen adalah sebagai berikut: agen 1 (agen merah) menempuh jarak 13.500, agen 2 (agen hijau) menempuh jarak 13.7117, agen 3 (agen biru) menempuh jarak 13.6195, agen 4 (agen cyan) menempuh jarak 11.6250, agen 5 (agen magenta) menempuh jarak 9.8494, and agen 6 (agen hitam) menempuh jarak 12.0208. Jadi akumulasi jarak yang ditempuh oleh semua agen adalah 74.3263.

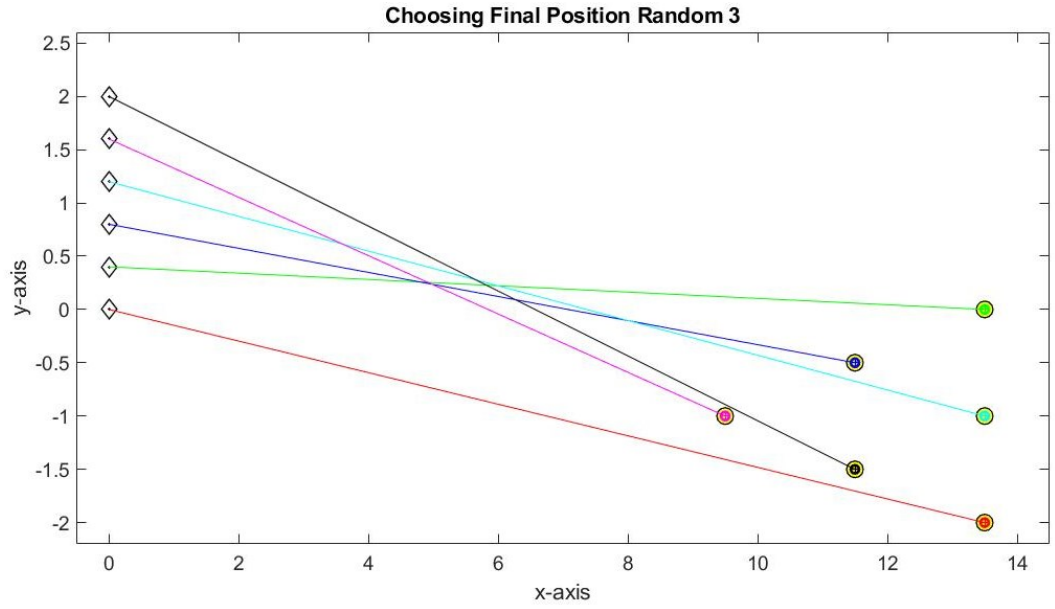


Gambar 4.6 Pemilihan Tujuan Akhir Acak 2

Dari Gambar 4.6, jarak tempuh dari masing-masing agen adalah sebagai berikut: agen 1 (agen merah) menempuh jarak 9.5525, agen 2 (agen hijau) menempuh jarak 13.7117, agen 3 (agen biru) menempuh jarak 13.5237, agen 4 (agen cyan) menempuh jarak 11.6250, agen 5 (agen magenta) perjalanan jarak 11.9105, dan agen 6 (agen hitam) menempuh jarak 13.8293. Jadi akumulasi jarak yang ditempuh oleh semua agen adalah 74.1526.

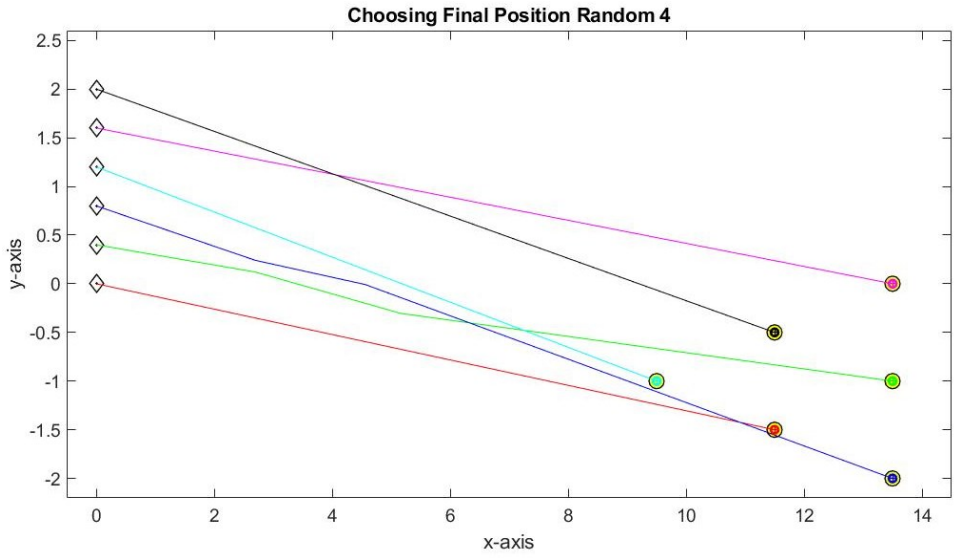
Dari Gambar 4.7, jarak tempuh dari masing-masing agen adalah sebagai berikut: agen 1 (agen merah) menempuh jarak 13.6473, agen 2 (agen hijau) menempuh jarak 13.5059, agen 3 (agen biru) menempuh jarak 11.5732, agen 4 (agen cyan) menempuh jarak 13.6781, agen 5 (agen magenta) menempuh jarak 9.8494, dan agen 6 (agen hitam) melakukan perjalanan jarak 12.0208. Jadi akumulasi jarak dijalani oleh semua agen adalah 74.2748.

Dari Gambar 4.8, jarak tempuh dari masing-masing agen adalah sebagai berikut: agen 1 (agen merah) menempuh jarak 11,5974, agen 2 (agen hijau) menempuh jarak 13,5724, agen 3 (agen biru) menempuh jarak 13.7873, agen 4 (agen cyan) menempuh jarak 9.7514, agen 5 (agen magenta) perjalanan jarak 13,5945, dan agen 6 (agen hitam) menempuh jarak 11.7686. Jadi akumulasi jarak yang ditempuh oleh semua agen adalah 74.0716.

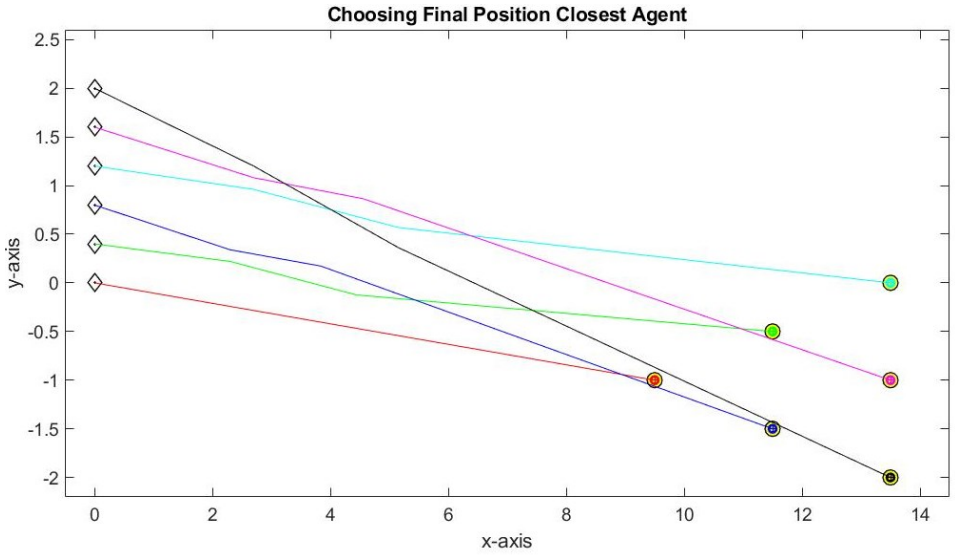


Gambar 4.7 Pemilihan Tujuan Akhir Acak 3

Dari keempat hasil pengujian titik tujuan acak yang telah dilakukan, akumulasi jarak yang ditempuh oleh seluruh agen masih belum menemukan hasil yang minimum. Hal ini dikarenakan hasil tujuan akhir yang masih diacak sehingga masih terdapat banyak kemungkinan pasangan agen dan posisi pada titik tujuan yang memiliki akumulasi jarak tempuh berbagai macam. Mungkin nantinya ada satu pasangan agen dan posisi pada titik tujuan yang memiliki nilai minimum dibandingkan dengan kemungkinan yang lainnya.



Gambar 4.8 Pemilihan Tujuan Akhir Acak 4

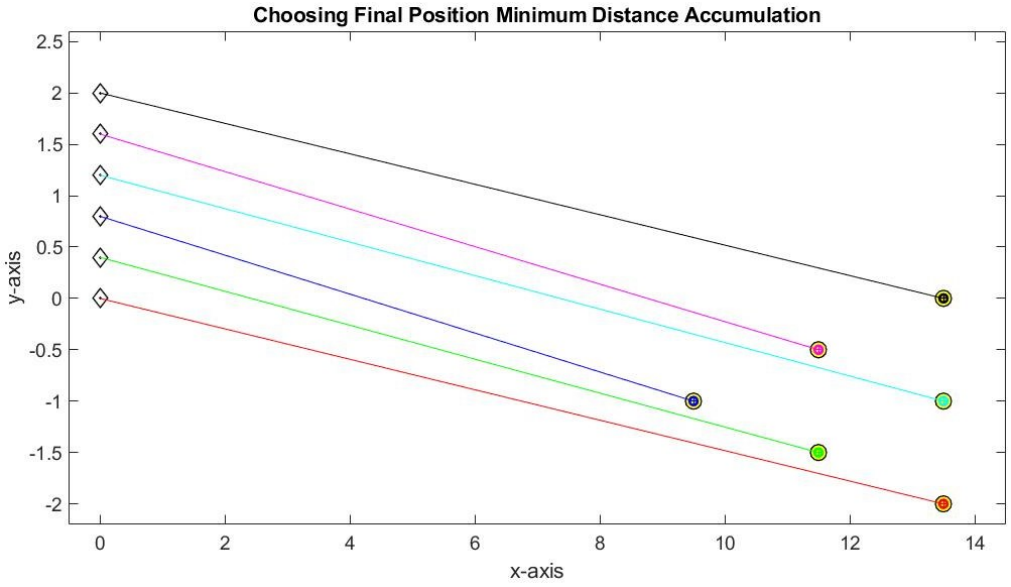


Gambar 4.9 Pemilihan Tujuan Akhir berdasarkan pada Agen yang Terdekat

Hasil pengujian untuk keadaan kedua ditunjukkan pada Gambar 4.9. Dari hasil pengujian tersebut, jarak tempuh dari masing-masing agen adalah sebagai berikut: agen 1 (agen merah) menempuh jarak 9.5525, agen 2 (agen hijau) menempuh jarak 11.5352, agen 3 (agen biru) menempuh jarak 11.7277, agen 4 (agen cyan) menempuh jarak 13,5532, agen 5 (agen magenta) menempuh jarak 13.7481, dan agen 6 (agen hitam) menempuh jarak 14.0801. Jadi akumulasi jarak yang ditempuh oleh semua agen adalah 74.1968.

Pengujian ini menunjukkan jika agen terdekat dengan posisi tujuan mengambil posisi itu sebagai posisi akhir, maka akan menghasilkan akumulasi jarak tempuh yang tidak minimum. Hal ini disebabkan oleh semakin besarnya jarak tempuh yang harus diambil oleh agen yang berada di urutan belakang karena jarak tempuh terpendek sudah diambil oleh agen yang berada di urutan depan.

Hasil pengujian untuk keadaan ketiga ditunjukkan pada Gambar 4.10. Dari hasil pengujian tersebut, jarak tempuh dari masing-masing agen adalah sebagai berikut: agen 1 (agen merah) menempuh jarak 13.6473, agen 2 (agen hijau) menempuh jarak 11.6559, agen 3 (agen biru) menempuh jarak 9.6690, agen 4 (agen cyan) menempuh jarak 13.6781, agen 5 (agen magenta) perjalanan jarak 11.6902, dan agen 6 (agen hitam) menempuh jarak 13.6473. Jadi akumulasi jarak yang ditempuh oleh semua agen adalah 73.9879.



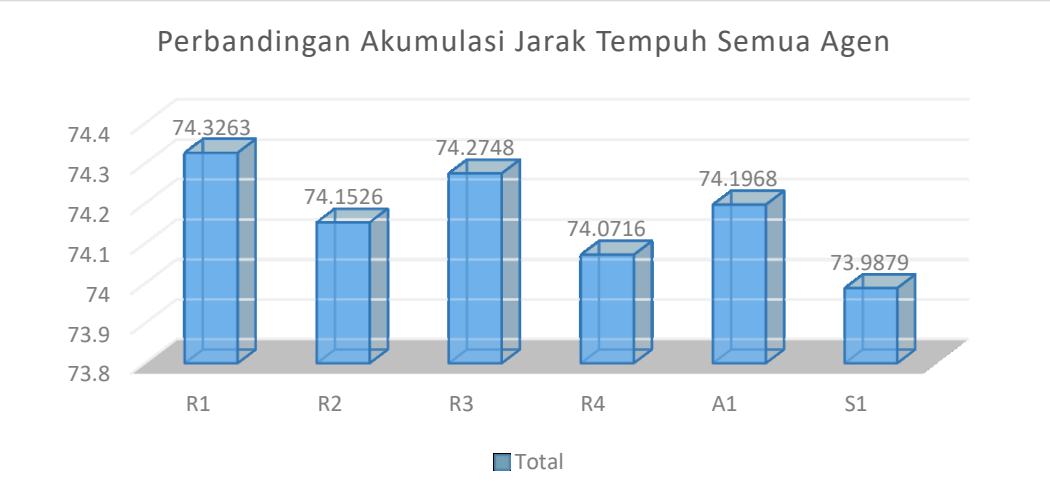
Gambar 4.10 Pemilihan Tujuan Akhir berdasarkan pada Kriteria

Algoritma yang diusulkan berhasil membuat agen mengambil posisi tujuan yang manghasilkan akumulasi jarak tempuh yang minimum. Dengan dihitungnya semua kemungkinan posisi tujuan yang diambil, lalu dipilihlah posisi tujuan dengan akumulasi jarak terpendek. Hal ini berhasil diwujudkan oleh algoritma yang disulkan.

Semua data jarak tempuh oleh masing-masing agen pada tiap kondisi dibandingkan pada Tabel 4.1. Data algoritma R1, R2, R3 dan R4 adalah simbol dari algoritma pemilihan tujuan akhir acak 1 hingga 4. Data algoritma A1 adalah simbol dari algoritma pemilihan tujuan akhir berdasarkan pada agen yang terdekat. Data algoritma S1 adalah simbol dari algoritma pemilihan posisi berdasarkan pada kriteria dari metode yang diusulkan. Data jarak D.1, D.2, D.3, D.4, D.5 dan D.6 adalah jarak yang ditempuh oleh masing-masing agen.

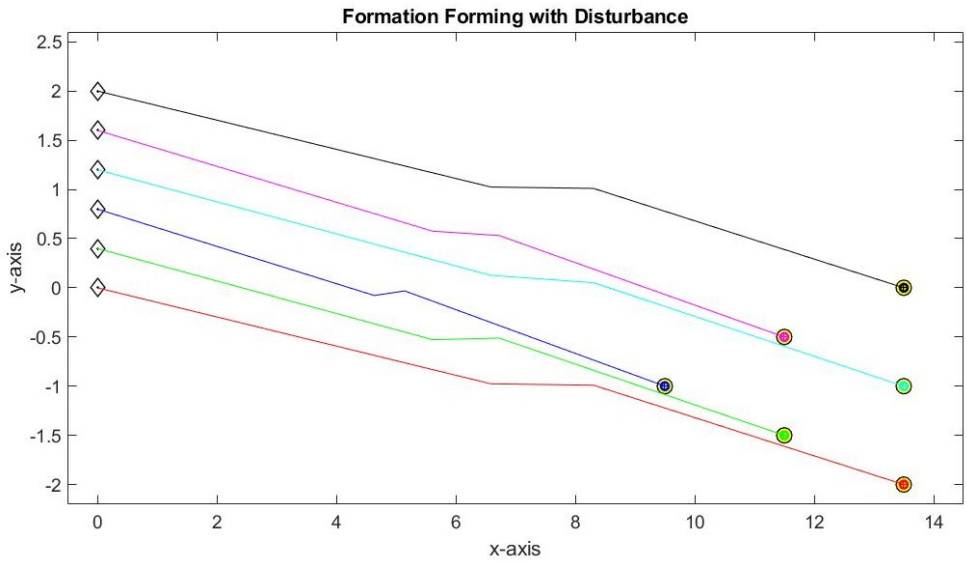
Tabel 4.1 Perbandingan Algoritma tiap Kondisi

Alg.	D.1	D.2	D.3	D.4	D.5	D.6	Total
R1	13.500	13.7117	13.6195	11.6250	9.8494	12.0208	74.3263
R2	9.5525	13.7117	13.5237	11.6250	11.9105	13.8293	74.1526
R3	13.6473	13.5059	11.5732	13.6781	9.8494	12.0208	74.2748
R4	11.5974	13.5724	13.7873	9.7514	13.5945	11.7686	74.0716
A1	9.5525	11.5352	11.7277	13.5532	13.7481	14.0801	74.1968
S1	13.6473	11.6559	9.6690	13.6781	11.6902	13.6473	**73.9879**



4.3 Pembentukan Formasi dengan Adanya Gangguan

Pengujian selanjutnya adalah dengan menambahkan gangguan saat semua agen bergerak menuju posisi akhir. Gangguan berupa hembusan angin yang dapat merubah arah laju semua agen. Pada pengujian ini angin berhembus dalam satu arah dan hanya terjadi pada saat tertentu. Besarnya gangguan angin $wind_d$ adalah $[-0.8 \quad 0.2]$. Kolom pertama untuk gangguan pada posisi x-*axis* dan kolom kedua untuk gangguan pada posisi y-*axis*. Gangguan angin terjadi pada saat iterasi ke-5 sampai ke-8. Hasil simulasi pengujian ditampilkan pada Gambar 4.11.



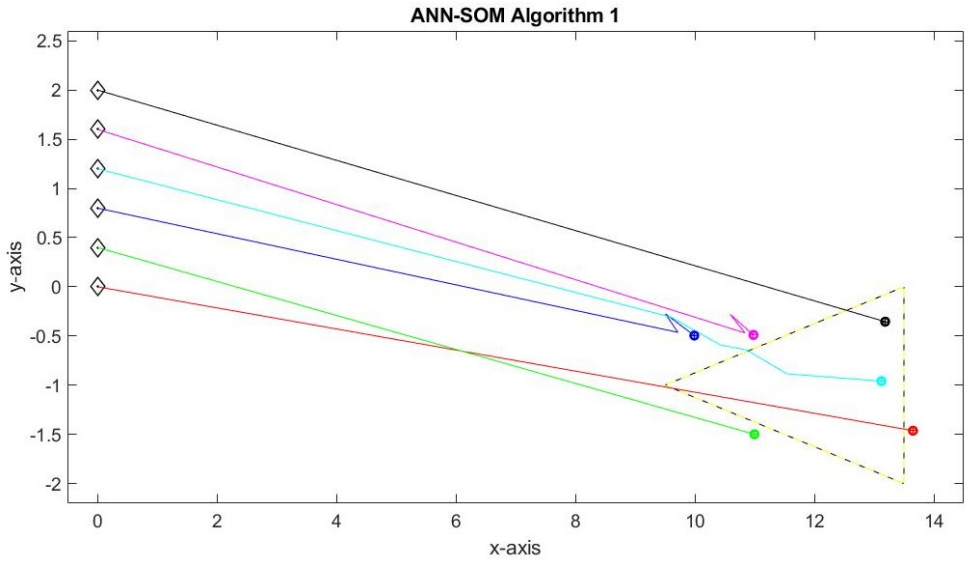
Gambar 4.11 Pembentukan Formasi dengan Adanya Gangguan

Akumulasi jarak yang ditempuh oleh semua agen adalah 74.1029. Akumulasi jarak ini lebih besar dari akumulasi jarak minimum yang didapat sebelumnya. Hal ini dikarenakan agen mengalami pembelokan oleh angin sehingga semua agen menempuh jalur yang lebih panjang dari sebelumnya.

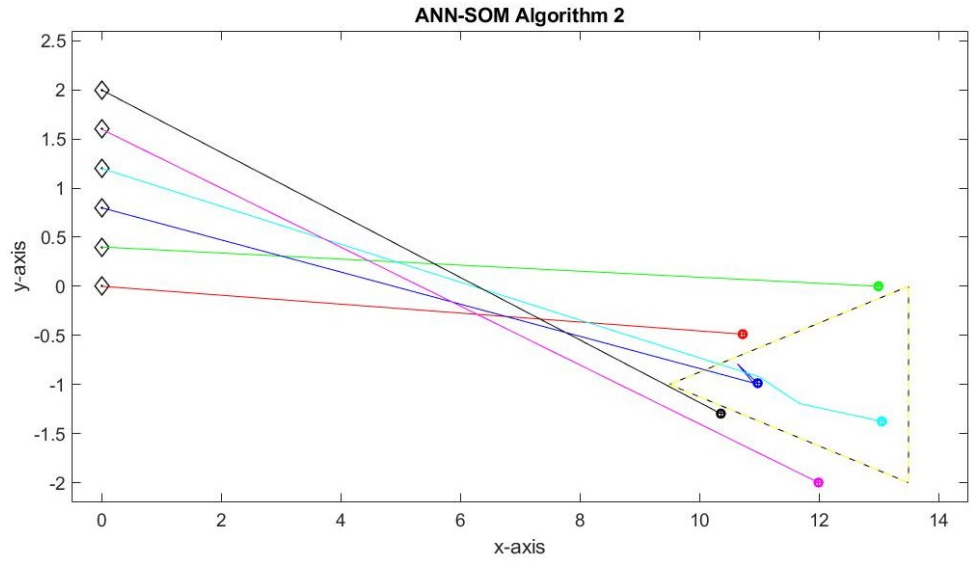
Agen berhasil mencapai posisi akhir setelah mengalami pembelokan jalur. Hal ini terjadi karena algoritma selalu menghitung posisi akhir untuk semua agen pada tiap iterasinya. Sehingga ketika agen keluar dari jalur optimalnya, maka algoritma akan menghitung jalur baru untuk mengarahkan agen pada posisi akhir dengan jarak minimum.

4.4 Pengujian Dibandingkan dengan Algoritma Original

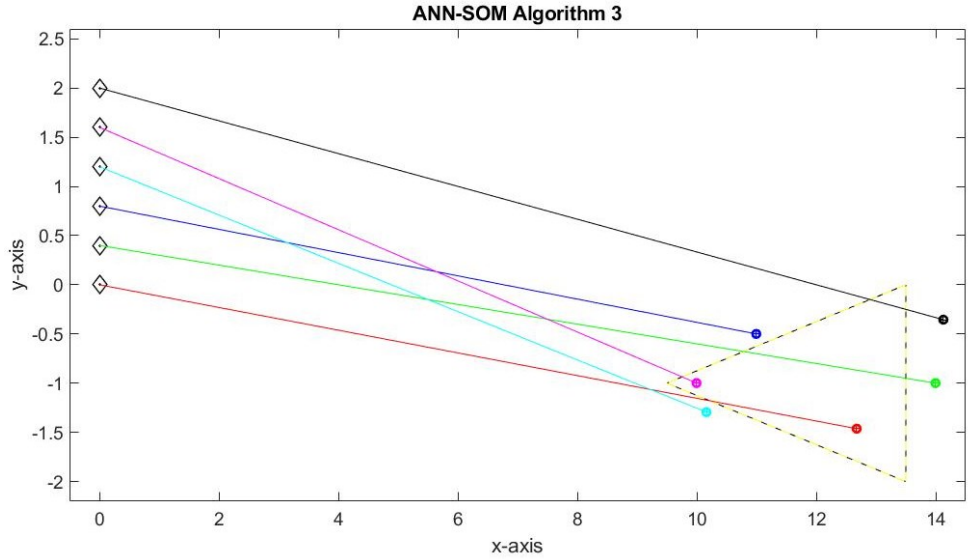
Algoritma yang penulis susun adalah algoritma baru dengan berdasarkan pada algoritma ANN-SOM yang sudah ada. Untuk membuktikan kelebihan algoritma yang penulis susun, maka perlu dilakukan perbandingan dengan algoritma ANN-SOM. Algoritma ANN-SOM akan dimodifikasi sesuai dengan situasi dalam penelitian ini. Sehingga hasil simulasi algoritma ANN-SOM sesuai dengan situasi pada penelitian ini ditunjukkan pada Gambar 4.12-15.



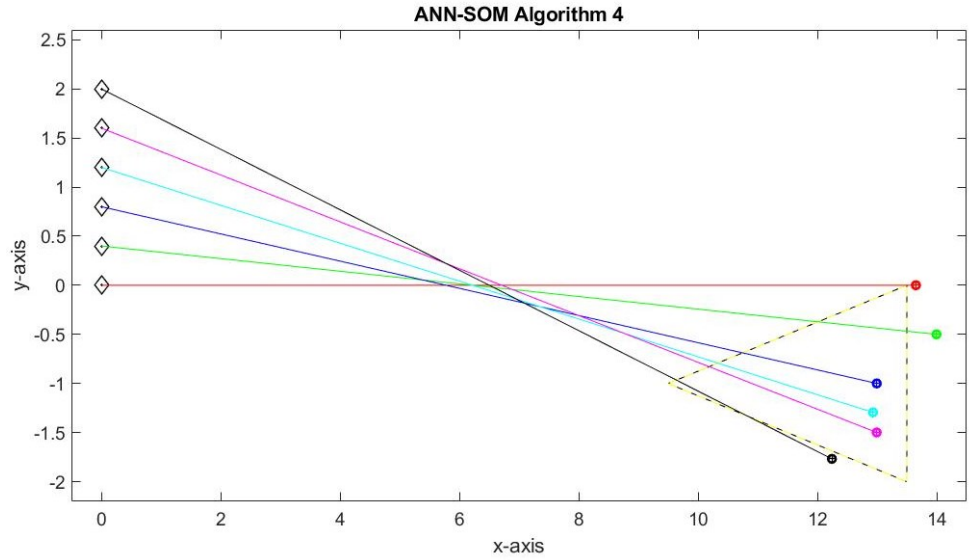
Gambar 4.12 Gerak Agen berdasarkan pada Algoritma ANN-SOM 1



Gambar 4.13 Gerak Agen berdasarkan pada Algoritma ANN-SOM 2



Gambar 4.14 Gerak Agen berdasarkan pada Algoritma ANN-SOM 3

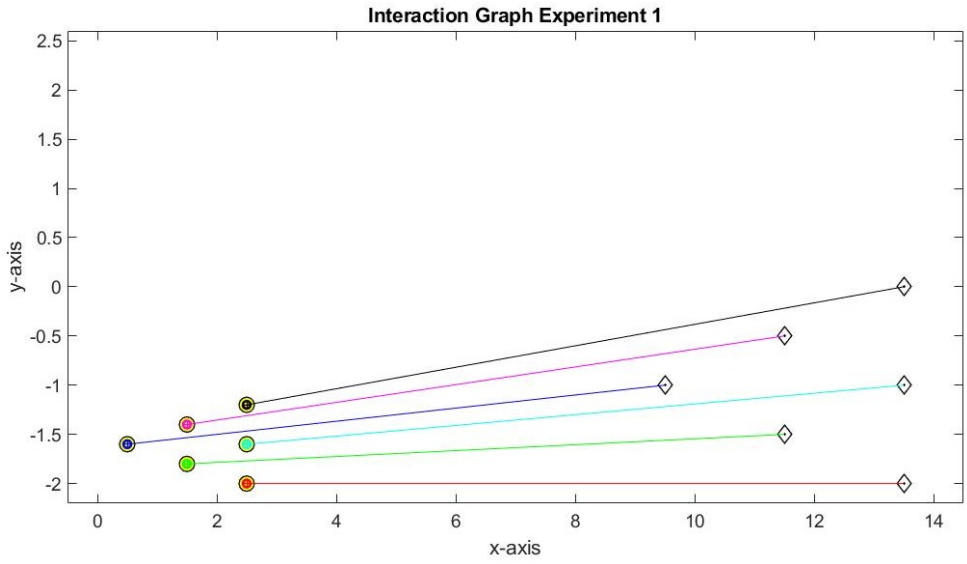


Gambar 4.15 Gerak Agen berdasarkan pada Algoritma ANN-SOM 4

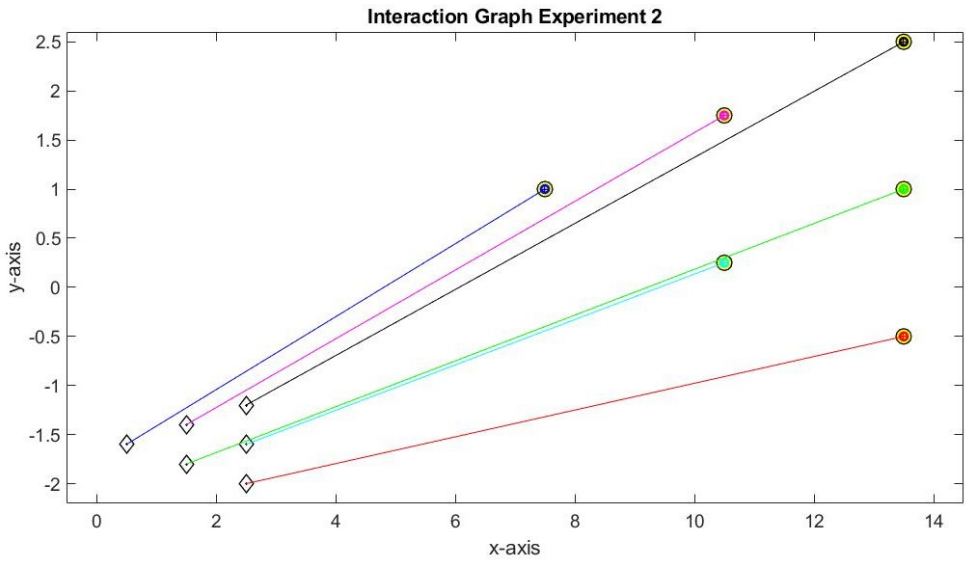
Dari 4 hasil simulasi seperti yang ditampilkan pada Gambar 4.12-15, dapat dilihat bahwa masih ada beberapa agen yang berhenti pada posisi didalam area segitiga. Hal ini menunjukkan bahwa algoritma ANN-SOM masih belum sanggup untuk menempatkan semua agen diluar area segitiga. Hal ini membuktikan bahwa algoritma yang penulis susun lebih baik dari algoritma ANN-SOM karena algoritma yang penulis susun berhasil menempatkan semua agen tepat di area segitiga.

4.5 Pembentukan Formasi dengan Interaksi Antar Agen

Pengujian dilakukan dengan melanjutkan simulasi untuk formasi yang sama dengan koordinat yang berbeda. Hasil simulasi ditunjukkan pada Gambar 4.16-17. Hasil simulasi menunjukkan, algoritma berhasil membentuk formasi berlanjutan dengan berdasar pada grafik interaksi antar agen. Hal ini menunjukkan pentingnya perumusan model interaksi antar agen untuk memudahkan dalam pembentukan formasi.



Gambar 4.16 Pembentukan Formasi Lanjut 1



Gambar 4.17 Pembentukan Formasi Lanjut 2

BAB 5
KESIMPULAN

5.1 Kesimpulan

Berdasar pada hasil simulasi yang telah dilakukan, algoritma yang diajukan telah berhasil menangani dua hal penting yaitu pembentukan formasi dan pemilihan posisi sesuai dengan kriteria.

Pada bagian pembentukan formasi, algoritma yang diajukan berhasil mengatur agen untuk bergerak menuju posisinya untuk membentuk formasi yang diinginkan. Dalam penelitian ini menggunakan tiga macam formasi yaitu formasi segitiga, persegi panjang dan lingkaran. Algoritma yang diusulkan berhasil membentuk ketiga formasi tersebut. Hal ini menunjukkan bahwa dengan berbagai macam formasi yang lain, algoritma yang diusulkan memiliki kemampuan untuk menanganinya.

Pada bagian pemilihan posisi sesuai dengan kriteria, algoritma yang diajukan berhasil memilih posisi tujuan yang menghasilkan akumulasi jarak tempuh agen adalah minimum yaitu 73.9879. Hal ini lalu dibandingkan dengan pemilihan posisi yang acak dan pemilihan posisi dengan agen terdekat. Terlihat jelas bahwa algoritma yang diusulkan memiliki keunggulan dibandingkan dengan algoritma yang lain.

5.2 Saran

Penelitian ini dapat dikembangkan dengan menambah algoritma kolaborasi antar agen dimana tiap agen memiliki sifat dan sikap yang berbeda antara satu dan yang lainnya sehingga akan menambah kontribusi untuk menuju kecerdasan buatan.

Penelitian pada ruang 3 dimensi menjadi salah satu pilihan yang menarik. Dengan pergerakan agen pada ruang 3 dimensi akan menambah tantangan dalam penyusunan jalur dan melewati jalur yang telah dibangun.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] J. M. Z. Maningo, G. E. U. Faelden, R. C. S. Nakano, A. A. Bandala, E. P. Dadios, "Obstacle Avoidance for Quadrotor Swarm using Artificial Neural Network Self-Organizing Map," in 8th IEEE International Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management (HNICEM), Cebu, 2015.

[2] K.-K. Oh, H.-S. Ahn, "Formation Control and Network Localization via Orientation Alignment," IEEE Transactions on Automatic Control, vol. 59, no. 2, February 2014.

[3] M. Aranda, G. L´opez-Nicol´as, C. Sag¨u´es, M. M. Zavlanos, "Distributed Formation Stabilization Using Relative Position Measurements in Local Coordinates," IEEE Transactions on Automatic Control, vol. 61, no. 12, December 2016.

[4] A. Sudaryanto, R. E. A. Kadier, A. Jazidie "Pengendalian Distribusi Quadrotor Swarm untuk Pelacakan Titik Centroid dengan Menggunakan Modified Artificial Neural Network Self Organizing Map (Modified Annsom)," Post-Graduate Thesis at Institut Teknologi Sepuluh Nopember, Surabaya, 2018.

[5] K.-K. Oh, M.-C. Park, H.-S. Ahn, "A Survey of Multi-agent Formation Control," Automatica, vol. 53, pages 424-440, March 2015.

[6] B. Santoso, P. Willy, " Metode Metaheuristik Konsep dan Implementasi," Guna Widya, Surabaya, 2001.

[7] Hornik, M. Stinchcombe, H. White, "Multilayer Feedforward Networks are Universal Approximators," Neural Networks, Volume 2, Issue 5, 1989, Pages 359-366, ISSN 0893-6080.

Halaman ini sengaja dikosongkan

LAMPIRAN

Program Utama

```
function [info, dataLog] = pswarm_coba3();
clc
close all
clear all

%% ===== Set the paramters =====
T=0.2; % Sampling Time
k=2; % Sampling counter

pi=[0 0   0   0   0   0
    0 0.4 0.8 1.2 1.6 2]; % the initial position
% pi=[13.5 11.5 9.5 13.5 11.5 13.5
%     -2   -1.5 -1  -1  -0.5  0]; % the initial position 2
% pi=[2.5 1.5  0.5  2.5  1.5  2.5
%     -2  -1.8 -1.6 -1.6 -1.4 -1.2]; % the initial position 3

x1(k-1)=pi(1,1); % initialize the state x1
y1(k-1)=pi(2,1); % initialize the state y1
x2(k-1)=pi(1,2); % initialize the state x2
y2(k-1)=pi(2,2); % initialize the state y2
x3(k-1)=pi(1,3); % initialize the state x3
y3(k-1)=pi(2,3); % initialize the state y3
x4(k-1)=pi(1,4); % initialize the state x4
y4(k-1)=pi(2,4); % initialize the state y4
x5(k-1)=pi(1,5); % initialize the state x5
y5(k-1)=pi(2,5); % initialize the state y5
x6(k-1)=pi(1,6); % initialize the state x6
y6(k-1)=pi(2,6); % initialize the state y6

% pf9=[13.5 11.5 9.5 13.5 11.5 13.5
%      -2   -1.5 -1  -1   -0.5 0]; % the fixed final position tiga
% pf9=[11.5 13.5 11.5 13.5 11.5 13.5
%      -2   -2   -1  -1    0    0]; % the fixed final position
balok
% pf9=[11.5 9.5  13.5 9.5  13.5 11.5
%      -2   -1.5 -1.5 -0.5 -0.5 0]; % the fixed final position
bundar

% pf7=[9.5  9.5  9.5  9.5  9.5  9.5
%     -1   -1   -1   -1   -1   -1]; % the final position

% pf7=[13.5 11.5  9.5 13.5 11.5 13.5
%      -2   -1.5  -1  -1   -0.5 0]; % the final position

% pf9=[2.5 1.5  0.5  2.5  1.5  2.5
%      -2  -1.8 -1.6 -1.6 -1.4 -1.2]; % the final position 2
% pf9=[13.5 10.5 7.5 13.5 10.5 13.5
%      -0.5 0.25 1.0 1.0  1.75 2.5]; % the final position 3
```

```
NN_wih=[1.0432 -1.3349 2.4346 2.6038 2.1960 2.3629]; % adjustment
from NN
% NN_v=abs(NN_wih - pi(2,:)); % adjust agent's speed
NN_v=[1 1 1 1 1 1];

% wind_d=[-0.8 0.2]; %wind disturbance
wind_d=[0 0];
min_d=5;
max_d=8;

gatabrak=[0.3 -0.2]; %avoid collision

sum11=0;

% theta(k-1)=0; % initialize the state theta

t=0; % intialize the time
tf=10; % final simulation time
%====================================

%% ===== Choosing position for achieving accumulative shortest
path via randomization =====
x34=[1 2 3 4 5 6];
out1=randperm(numel(x34),6);
% out1=[2 4 1 3 6 5];

% pf7=[pf9(:,out1(1)) pf9(:,out1(2)) pf9(:,out1(3)) pf9(:,out1(4))
pf9(:,out1(5)) pf9(:,out1(6))];
%====================================

%% ===== Choosing position for achieving accumulative shortest
path via shortest distance =====
selisih1 = bsxfun(@minus, pf9, pi(:,1));
selisih2 = bsxfun(@minus, pf9, pi(:,2));
selisih3 = bsxfun(@minus, pf9, pi(:,3));
selisih4 = bsxfun(@minus, pf9, pi(:,4));
selisih5 = bsxfun(@minus, pf9, pi(:,5));
selisih6 = bsxfun(@minus, pf9, pi(:,6));

d1 = sqrt(sum(selisih1.^2));
d2 = sqrt(sum(selisih2.^2));
d3 = sqrt(sum(selisih3.^2));
d4 = sqrt(sum(selisih4.^2));
d5 = sqrt(sum(selisih5.^2));
d6 = sqrt(sum(selisih6.^2));

[min11,I11] = min(d1);
[min12,I12] = min(d2);
[min13,I13] = min(d3);
[min14,I14] = min(d4);
[min15,I15] = min(d5);
[min16,I16] = min(d6);

[min17,I17] = min([min11 min12 min13 min14 min15 min16]);
```

```
[min21,I21] = mink(d1,2);
[min22,I22] = mink(d2,2);
[min23,I23] = mink(d3,2);
[min24,I24] = mink(d4,2);
[min25,I25] = mink(d5,2);
[min26,I26] = mink(d6,2);

[min27,I27] = min([100 min22(2) min23(2) min24(2) min25(2)
min26(2)]);

[min31,I31] = mink(d1,3);
[min32,I32] = mink(d2,3);
[min33,I33] = mink(d3,3);
[min34,I34] = mink(d4,3);
[min35,I35] = mink(d5,3);
[min36,I36] = mink(d6,3);

[min37,I37] = min([100 100 min33(3) min34(3) min35(3) min36(3)]);

[min41,I41] = mink(d1,4);
[min42,I42] = mink(d2,4);
[min43,I43] = mink(d3,4);
[min44,I44] = mink(d4,4);
[min45,I45] = mink(d5,4);
[min46,I46] = mink(d6,4);

[min47,I47] = min([100 100 100 min44(4) min45(4) min46(4)]);

[min51,I51] = mink(d1,5);
[min52,I52] = mink(d2,5);
[min53,I53] = mink(d3,5);
[min54,I54] = mink(d4,5);
[min55,I55] = mink(d5,5);
[min56,I56] = mink(d6,5);

[min57,I57] = min([100 100 100 100 min55(5) min56(5)]);

[min61,I61] = mink(d1,6);
[min62,I62] = mink(d2,6);
[min63,I63] = mink(d3,6);
[min64,I64] = mink(d4,6);
[min65,I65] = mink(d5,6);
[min66,I66] = mink(d6,6);

[min67,I67] = min([100 100 100 100 100 min66(6)]);

% pf7=[pf9(:,I11(1)) pf9(:,I22(2)) pf9(:,I33(3)) pf9(:,I44(4))
pf9(:,I55(5)) pf9(:,I66(6))];
%=====================================

%% ===== Allocate memory for the dataLog =====
% maxIter = (tf-t)/T;
% dataLog(1) = makeStruct(x1,x2,x3,x4,x5,x6, y1,y2,y3,y4,y5,y6);
% info.iter = 1:maxIter;
%=====================================
```

```
%% ========== The main loop ==========
tic
while(t<=tf)
t=t+T; % increase the time

pp=[x1(k-1) x2(k-1) x3(k-1) x4(k-1) x5(k-1) x6(k-1)
    y1(k-1) y2(k-1) y3(k-1) y4(k-1) y5(k-1) y6(k-1)];

%===== Choosing position for achieving accumulative shortest path
via permutation =====
x35=[1 2 3 4 5 6];
out2=perms(x35);
for i = 1:720
    pf720(:,:,i)=[pf9(:,out2(i,1)) pf9(:,out2(i,2))
pf9(:,out2(i,3)) pf9(:,out2(i,4)) pf9(:,out2(i,5))
pf9(:,out2(i,6))];
end

selisih720 = bsxfun(@minus, pf720, pp);
d720 = sqrt(sum(selisih720.^2));
sum720 = sum(d720);
[min720,I720] = min(sum720);

pf7=[pf9(:,out2(I720,1)) pf9(:,out2(I720,2)) pf9(:,out2(I720,3))
pf9(:,out2(I720,4)) pf9(:,out2(I720,5)) pf9(:,out2(I720,6))];
%=====================================

Vx1=pf7(1,1)-x1(k-1);
Vy1=pf7(2,1)-y1(k-1);
Vx2=pf7(1,2)-x2(k-1);
Vy2=pf7(2,2)-y2(k-1);
Vx3=pf7(1,3)-x3(k-1);
Vy3=pf7(2,3)-y3(k-1);
Vx4=pf7(1,4)-x4(k-1);
Vy4=pf7(2,4)-y4(k-1);
Vx5=pf7(1,5)-x5(k-1);
Vy5=pf7(2,5)-y5(k-1);
Vx6=pf7(1,6)-x6(k-1);
Vy6=pf7(2,6)-y6(k-1);

% W=0;
% theta(k)=W*T+theta(k-1);

%===== Wind disturbance's model =====
if (k >= min_d) && (k <= max_d)
    disturbance = wind_d;
else
    disturbance = [0 0];
end
%=====================================

%===== Collition's avoidance's model =====
if (sqrt(((x2(k-1)-x1(k-1)).^2)+((y2(k-1)-y1(k-1)).^2)) <= 0.2)
    geser1 = gatabrak;
else
```

```
    geser1 = [0 0];
end

if (sqrt(((x1(k-1)-x2(k-1)).^2)+((y1(k-1)-y2(k-1)).^2)) <= 0.2) ||
(sqrt(((x3(k-1)-x2(k-1)).^2)+((y3(k-1)-y2(k-1)).^2)) <= 0.2)
    geser2 = gatabrak;
else
    geser2 = [0 0];
end

if (sqrt(((x2(k-1)-x3(k-1)).^2)+((y2(k-1)-y3(k-1)).^2)) <= 0.2) ||
(sqrt(((x4(k-1)-x3(k-1)).^2)+((y4(k-1)-y3(k-1)).^2)) <= 0.2)
    geser3 = gatabrak;
else
    geser3 = [0 0];
end

if (sqrt(((x3(k-1)-x4(k-1)).^2)+((y3(k-1)-y4(k-1)).^2)) <= 0.2) ||
(sqrt(((x5(k-1)-x4(k-1)).^2)+((y5(k-1)-y4(k-1)).^2)) <= 0.2)
    geser4 = gatabrak;
else
    geser4 = [0 0];
end

if (sqrt(((x4(k-1)-x5(k-1)).^2)+((y4(k-1)-y5(k-1)).^2)) <= 0.2) ||
(sqrt(((x6(k-1)-x5(k-1)).^2)+((y6(k-1)-y5(k-1)).^2)) <= 0.2)
    geser5 = gatabrak;
else
    geser5 = [0 0];
end

if (sqrt(((x5(k-1)-x6(k-1)).^2)+((y5(k-1)-y6(k-1)).^2)) <= 0.2)
    geser6 = gatabrak;
else
    geser6 = [0 0];
end
%====================================

x1(k)=Vx1*T*NN_v(1)+x1(k-1)+disturbance(1)-geser1(1); %
calculating x1
y1(k)=Vy1*T*NN_v(1)+y1(k-1)+disturbance(2)-geser1(2); %
calculating y1
x2(k)=Vx2*T*NN_v(2)+x2(k-1)+disturbance(1)+geser2(1); %
calculating x2
y2(k)=Vy2*T*NN_v(2)+y2(k-1)+disturbance(2)+geser2(2); %
calculating y2
x3(k)=Vx3*T*NN_v(3)+x3(k-1)+disturbance(1)-geser3(1); %
calculating x3
y3(k)=Vy3*T*NN_v(3)+y3(k-1)+disturbance(2)-geser3(2); %
calculating y3
x4(k)=Vx4*T*NN_v(4)+x4(k-1)+disturbance(1)+geser4(1); %
calculating x4
y4(k)=Vy4*T*NN_v(4)+y4(k-1)+disturbance(2)+geser4(2); %
calculating y4
x5(k)=Vx5*T*NN_v(5)+x5(k-1)+disturbance(1)-geser5(1); %
calculating x5
```

```
y5(k)=Vy5*T*NN_v(5)+y5(k-1)+disturbance(2)-geser5(2); %
calculating y5
x6(k)=Vx6*T*NN_v(6)+x6(k-1)+disturbance(1)+geser6(1); %
calculating x6
y6(k)=Vy6*T*NN_v(6)+y6(k-1)+disturbance(2)+geser6(2); %
calculating y6

selisih10=[x1(k)-x1(k-1) x2(k)-x2(k-1) x3(k)-x3(k-1) x4(k)-x4(k-1)
x5(k)-x5(k-1) x6(k)-x6(k-1)
           y1(k)-y1(k-1) y2(k)-y2(k-1) y3(k)-y3(k-1) y4(k)-y4(k-1)
y5(k)-y5(k-1) y6(k)-y6(k-1)];
d10 = sqrt(sum(selisih10.^2));
sum10 = sum(d10);
sum11=sum(d10)+sum11

draw_robot(); % Draw the robot and it's path
k=k+1; % increase the sampling counter
end
toc
%=====================================

%% === Draw the mobile robot & Path ====
function draw_robot()
xmin=-0.5; % setting the figure limits
xmax=14.5;
ymin=-2.2;
ymax=2.6;

plot(pi(1,:),pi(2,:),'kd','linewidth',5) % Drawing the Initial
Position
hold on

plot(pi(1,:),pi(2,:),'wd','linewidth',3) % Drawing the Initial
Position
hold on

plot(pf9(1,:),pf9(2,:),'ko','linewidth',7) % Drawing the Final
Position
hold on

plot(pf9(1,:),pf9(2,:),'yo','linewidth',5) % Drawing the Final
Position
axis([xmin xmax ymin ymax]) % setting the figure limits
set(gcf,'units','points','position',[50,50,900,450])
hold on

plot(x1,y1,'-r') % Drawing the Path
plot(x2,y2,'-g') % Drawing the Path
plot(x3,y3,'-b') % Drawing the Path
plot(x4,y4,'-c') % Drawing the Path
plot(x5,y5,'-m') % Drawing the Path
plot(x6,y6,'-k') % Drawing the Path
hold on

% Body for each agen
v1=[pi(1,1);pi(2,1)];
```

```
v2=[pi(1,2);pi(2,2)];
v3=[pi(1,3);pi(2,3)];
v4=[pi(1,4);pi(2,4)];
v5=[pi(1,5);pi(2,5)];
v6=[pi(1,6);pi(2,6)];

% R=[cos(theta(k)) -sin(theta(k));sin(theta(k)) cos(theta(k))]; %
Rotation Matrix
% R=[1];
P1=[x1(k);y1(k)]; % Position Matrix
P2=[x2(k);y2(k)]; % Position Matrix
P3=[x3(k);y3(k)]; % Position Matrix
P4=[x4(k);y4(k)]; % Position Matrix
P5=[x5(k);y5(k)]; % Position Matrix
P6=[x6(k);y6(k)]; % Position Matrix

v1=P1;
v2=P2;
v3=P3;
v4=P4;
v5=P5;
v6=P6;

% Plot body for each agen
plot(v1(1,:),v1(2,:),'ro','linewidth',2);
plot(v1(1,:),v1(2,:),'r+','linewidth',1);
plot(v2(1,:),v2(2,:),'go','linewidth',2);
plot(v2(1,:),v2(2,:),'g+','linewidth',1);
plot(v3(1,:),v3(2,:),'bo','linewidth',2);
plot(v3(1,:),v3(2,:),'b+','linewidth',1);
plot(v4(1,:),v4(2,:),'co','linewidth',2);
plot(v4(1,:),v4(2,:),'c+','linewidth',1);
plot(v5(1,:),v5(2,:),'mo','linewidth',2);
plot(v5(1,:),v5(2,:),'m+','linewidth',1);
plot(v6(1,:),v6(2,:),'ko','linewidth',2);
plot(v6(1,:),v6(2,:),'k+','linewidth',1);

title('Agents, Initial Positions, and Destination Positions');
% title('Formation Forming of Triangle Formation');
% title('Formation Forming of Rectangle Formation');
% title('Formation Forming of Circle Formation');
% title('Choosing Final Position Random 1');
% title('Choosing Final Position Random 2');
% title('Choosing Final Position Random 3');
% title('Choosing Final Position Random 4');
% title('Choosing Final Position Closest Agent');
% title('Choosing Final Position Minimum Distance Accumulation');
% title('Formation Forming with Disturbance');
% title('Interaction Graph Experiment 1');
% title('Interaction Graph Experiment 2');
xlabel('x-axis');
ylabel('y-axis');
ax = gca;
ax.FontSize = 14;

drawnow
% pause(0.9)
```

```
hold off
end
%====================================

%% ===== Distance calculation =====
selisih8 = bsxfun(@minus, pf7, pi);
d8 = sqrt(sum(selisih8.^2))

sum8 = sum(d8)

selisih9 = bsxfun(@minus, pf9, pi);
d9 = sqrt(sum(selisih9.^2))

sum9 = sum(d9)
%====================================

%% ===== Write memory to dataLog =====
% iter = 1:maxIter;
% dataLog(1) = makeStruct(x1,x2,x3,x4,x5,x6, y1,y2,y3,y4,y5,y6);
% info.pf = pf1;
% info.selisih = selisih;
% info.d1 = d1;
% info.closest = closest;
% info.akumulasi = akumulasi;
%====================================

end
```

Program Koreksi Sudut

```
clc
close all
clear all

pf9 = [13.5 11.5 9.5 13.5 11.5 13.5
       -2   -1.5 -1  -1   -0.5 0];

% agen1 = [pf9(1,3) pf9(2,3)];
% agen2 = [pf9(1,5) pf9(2,5)];
% agen3 = [pf9(1,2) pf9(2,2)];
% agen4 = [pf9(1,6) pf9(2,6)];
% agen5 = [pf9(1,4) pf9(2,4)];
% agen6 = [pf9(1,1) pf9(2,1)];

g1x = [pf9(1,3) pf9(1,1)]; % membuat garis antara agen 1 dan 6
g1y = [pf9(2,3) pf9(2,1)];
line(g1x,g1y)

g2x = [pf9(1,3) pf9(1,6)]; % membuat garis antara agen 1 dan 4
g2y = [pf9(2,3) pf9(2,6)];
line(g2x,g2y)

g3x = [pf9(1,6) pf9(1,1)]; % membuat garis antara agen 4 dan 6
g3y = [pf9(2,6) pf9(2,1)];
```

```
line(g3x,g3y)

g4x = [pf9(1,5) pf9(1,2)]; % membuat garis antara agen 2 dan 3
g4y = [pf9(2,5) pf9(2,2)];
line(g4x,g4y,'Color','r')

g5x = [pf9(1,5) pf9(1,4)]; % membuat garis antara agen 2 dan 5
g5y = [pf9(2,5) pf9(2,4)];
line(g5x,g5y,'Color','r')

g6x = [pf9(1,4) pf9(1,2)]; % membuat garis antara agen 5 dan 3
g6y = [pf9(2,4) pf9(2,2)];
line(g6x,g6y,'Color','r')
hold on

xmin=8.5; % setting the figure limits
xmax=14.5;
ymin=-2.5;
ymax=0.5;

axis([xmin xmax ymin ymax]) % setting the figure limits
% set(gcf,'units','points','position',[50,50,900,450])

sudut1 = atan((g1y(2)-g1y(1))/(g1x(2)-g1x(1)))*180/pi % sudut
antara garis g1 dengan sumbu x
sudut2 = atan((g2y(2)-g2y(1))/(g2x(2)-g2x(1)))*180/pi % sudut
antara garis g2 dengan sumbu x
sudut3 = atan((g3y(2)-g3y(1))/(g3x(2)-g3x(1)))*180/pi % sudut
antara garis g3 dengan sumbu x
sudut4 = atan((g4y(2)-g4y(1))/(g4x(2)-g4x(1)))*180/pi % sudut
antara garis g4 dengan sumbu x
sudut5 = atan((g5y(2)-g5y(1))/(g5x(2)-g5x(1)))*180/pi % sudut
antara garis g5 dengan sumbu x
sudut6 = atan((g6y(2)-g6y(1))/(g6x(2)-g6x(1)))*180/pi % sudut
antara garis g6 dengan sumbu x

sagen1 = abs(sudut1)+abs(sudut2)
sagen4 = abs(sudut3)-abs(sudut2)
sagen6 = abs(sudut3)-abs(sudut1)
sagen2 = abs(sudut4)-abs(sudut5)
sagen3 = abs(sudut4)-abs(sudut6)
sagen5 = abs(sudut5)+abs(sudut6)

plot(pf9(1,:),pf9(2,:),'ko','linewidth',7) % Drawing the Final
Position
hold on

plot(pf9(1,:),pf9(2,:),'yo','linewidth',5) % Drawing the Final
Position

title('Simulasi Grafik Interaksi');
xlabel('x-axis');
ylabel('y-axis');
ax = gca;
ax.FontSize = 14;
```

Program NN-Back Propagation Algorithm

```
%-------------------------------------------------------------
% MATLAB neural network backprop code
% by Phil Brierley
% www.philbrierley.com
% 29 March 2006
%
% This code implements the basic backpropagation of
% error learning algorithm.
%
% adjust the learning rate with the slider
%
% feel free to improve!
%
%-------------------------------------------------------------

%user specified values
hidden_neurons = 2;
epochs = 1000;

% ------- load in the data -------

% XOR data
train_inp = [1 1; 1 0; 0 1; 0 0];
train_out = [1; 0; 0; 1];

% check same number of patterns in each
if size(train_inp,1) ~= size(train_out,1)
    disp('ERROR: data mismatch')
   return
end

%standardise the data to mean=0 and standard deviation=1
%inputs
mu_inp = mean(train_inp);
sigma_inp = std(train_inp);
train_inp = (train_inp(:,:) - mu_inp(:,1)) / sigma_inp(:,1);

%outputs
train_out = train_out';
mu_out = mean(train_out);
sigma_out = std(train_out);
train_out = (train_out(:,:) - mu_out(:,1)) / sigma_out(:,1);
train_out = train_out';

%read how many patterns
patterns = size(train_inp,1);

%add a bias as an input
bias = ones(patterns,1);
train_inp = [train_inp bias];

%read how many inputs
inputs = size(train_inp,2);
```

```
%---------- data loaded ------------

%--------- add some control buttons ---------

%add button for early stopping
hstop = uicontrol('Style','PushButton','String','Stop',
'Position', [5 5 70 20],'callback','earlystop = 1;');
earlystop = 0;

%add button for resetting weights
hreset = uicontrol('Style','PushButton','String','Reset Wts',
'Position', get(hstop,'position')+[75 0 0 0],'callback','reset =
1;');
reset = 0;

%add slider to adjust the learning rate
hlr =
uicontrol('Style','slider','value',.1,'Min',.01,'Max',1,'SliderSte
p',[0.01 0.1],'Position', get(hreset,'position')+[75 0 100 0]);

% ---------- set weights -----------------
%set initial random weights
% weight_input_hidden = (randn(inputs,hidden_neurons) - 0.5)/10;
% weight_hidden_output = (randn(1,hidden_neurons) - 0.5)/10;
weight_input_hidden = [0   0.4
                       0.8 1.2
                       1.6 2];
weight_hidden_output = [0 1];

%-----------------------------------
%--- Learning Starts Here! ---------
%-----------------------------------

%do a number of epochs
for iter = 1:epochs

    %get the learning rate from the slider
    alr = get(hlr,'value');
    blr = alr / 10;

    %loop through the patterns, selecting randomly
    for j = 1:patterns

        %select a random pattern
        patnum = round((rand * patterns) + 0.5);
        if patnum > patterns
            patnum = patterns;
        elseif patnum < 1
            patnum = 1;
        end

        %set the current pattern
        this_pat = train_inp(patnum,:);
        act = train_out(patnum,1);
```

```
        %calculate the current error for this pattern
        hval = (tansig(this_pat*weight_input_hidden))';
        pred = tansig(hval'*weight_hidden_output');
        error = pred - act;

        % adjust weight hidden - output
        delta_HO = error.*blr.*hval;
        weight_hidden_output = weight_hidden_output - delta_HO';

        % adjust the weights input - hidden
        delta_IH= alr.*error.*weight_hidden_output'.*(1-
(hval.^2))*this_pat;
        weight_input_hidden = weight_input_hidden - delta_IH';

    end
    % -- another epoch finished

    %plot overall network error at end of each epoch
    pred =
weight_hidden_output*tanh(train_inp*weight_input_hidden)';
    error = pred' - train_out;
    err(iter) =  (sum(error.^2))^0.5;

    figure(1);
    plot(err)

        %reset weights if requested
    if reset
        weight_input_hidden = (randn(inputs,hidden_neurons) -
0.5)/10;
        weight_hidden_output = (randn(1,hidden_neurons) - 0.5)/10;
        fprintf('weights reaset after %d epochs\n',iter);
        reset = 0;
    end

    %stop if requested
    if earlystop
        fprintf('stopped at epoch: %d\n',iter);
        break
    end

    %stop if error is small
    if err(iter) < 0.001
        fprintf('converged at epoch: %d\n',iter);
        break
    end

end

   %-----FINISHED---------
   %display actual,predicted & error
   fprintf('state after %d epochs\n',iter);
   a = (train_out* sigma_out(:,1)) + mu_out(:,1);
   b = (pred'* sigma_out(:,1)) + mu_out(:,1);
   act_pred_err = [a b b-a]
```

BIODATA PENULIS

Bayu Fandidarma dilahirkan di Madiun pada tanggal 22 Juni 1990. Putra Pertama dari Bapak Sudarmadi dan Ibu Sukartiningsih. Penulis menempuh pendidikan formalnya dimulai di SDN Madiun Lor 06 Madiun, SMPN 1 Madiun, SMAN 2 Madiun, dan kemudian menempuh pendidikan tingginya di Teknik Elektro, **Universitas Gadjah Mada** Yogyakarta pada bulan September 2008. Pada bulan Januari 2013, penulis berhasil menyelesaikan pendidikan sarjananya dan mendapatkan gelar **Sarjana Teknik (ST.)**. Penulis melanjutkan pendidikan magister di Departemen Teknik Elektro, **Institut Teknologi Sepuluh Nopember** Surabaya dengan mengambil Bidang Keahlian Teknik Sistem Pengaturan pada bulan Januari 2017. Pada bulan Februari 2019, penulis berhasil menyelesaikan pendidikan magisternya dan mendapatkan gelar **Magister Teknik (MT.)**.

Penulis

bayufandidarma@gmail.com