Kontrol Diskrit Hexapod dengan Metode Elman Neural Network dengan Studi Kasus Driving Simulator Game Sederhana

Lucky Leonardo Salim[1], Gregorius Satia Budhi[2], Felix Pasila[3]
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-mail: m26412031@john.petra.ac.id[1], greg@petra.ac.id[2], felix@petra.ac.id[3]

ABSTRAK

Permainan adalah salah satu cara untuk mendapatkan hiburan dan untuk menghilangkan rasa jenuh. Salah satu kategori dari permainan adalah permainan simulasi dimana pemain dapat seolah-olah merasakan efek dari jalannya permainan. Salah satu alat yang dapat digunakan untuk membantu memperoleh efek simulasi adalah Diskrit Hexapod.

Desain Diskrit Hexapod dapat digunakan untuk *Driving Simulator* dengan Arduino sebagai otak dan pengontrol untuk Diskrit Hexapod. Diskrit Hexapod dapat bergerak sesuai dengan tampilan visual dari gerakkan mobil pada *Driving Simulator*. Diskrit Hexapod akan bergerak secara akurat apabila menerima masukkan dengan nilai sudut x mulai dari -20° sampai 20° dan nilai sudut z mulai dari dari -20° sampai 20° dan nilai sudut y mulai dari -90° sampai 90°. Pasangan data yang digunakan sebagai masukkan program berjumlah 161 pasang dengan hasil rata-rata error RMSE 0,0632.

Kata Kunci : Jaringan Saraf Tiruan, Backpropagation, *Driving Simulator, Arduino, Diskrit Hexapod*.

ABSTRACT

Game is one of many ways to get entertainment and to relief from stress. One of many category of game is simulation game where player can feel the effect of the gameplay. One of the many instrument that can help to get the effect of simulation is Diskrit Hexapod.

The design of Diskrit Hexapod can be used in Driving Simulator with Arduino as its brain and controller for the Diskrit Hexapod. Diskrit Hexapod can move according to the visual display of the car in Driving Simulator game. Diskrit Hexapod can move accurately if receive the input of x-axis and z-axis from range -20° to 20° and y-axis from range -90° to 90°. The data used for the program is made of 161 pairs of inputs and has trained root-mean-squared-error of 0,0632

Keywords : *Artificial Neural Network*, *Backpropagation, Driving Simulator, Arduino, Diskrit Hexapod.*

1. PENDAHULUAN

Perkembangan teknologi di Indonesia setara dengan perkembangan jaman yang semakin lama semakin maju, perkembangan teknologi juga dipengaruhi dengan banyaknya inovasi dan penemuan yang unik. Teknologi dapat dimanfaatkan dalam berbagai hal seperti membantu kehidupan manusia sehari-hari dan membantu dalam perkembangan dan penelitian untuk membandingkan metode atau teori yang telah ada.

Diskrit Hexapod merupakan salah satu teknologi hasil penelitian Universitas Kristen Petra yang masih dikembangkan, Diskrit Hexapod berupa platform MA^{3}I yang terbentuk dari kerangka mekanikal yang dikombinasikan dengan kursi sehingga memiliki fitur untuk menghasilkan suatu efek goyang ketika menerima suatu *input* data tertentu [6, 7]. Efek goyang ini dikarenakan Diskrit Hexapod menggunakan kerangka mekanikal berupa diskrit valve yang akan menghasilkan gerakan memanjang dan memendek secara pararel. Diskrit valve hanya memiliki dua status yaitu memanjang dan memendek sehingga gerakan yang dihasilkan akan terlihat kaku.

Sistem modul yang digunakan untuk mengontrol pergerakan Diskrit Hexapod sampai saat ini adalah dengan menggunakan data tabel yang berisi data dari fuzzy rule, data tabel ini nanti akan digunakan sebagai penentu kondisi status untuk Diksrit Hexapod setelah menerima input.

Sistem modul yang menggunakan data tabel dari fuzzy rule ini memiliki beberapa kelemahan yaitu apabila jumlah kondisi semakin banyak maka akan memperlambat kerja modul, oleh karena itu akan dilakukan penelitian lanjutan untuk pembuatan modul dengan menggunakan metode jaringan saraf tiruan dengan Elman Recurrent Neural Network.

Jaringan saraf tiruan memiliki kekurangan dan kelebihan apabila dibandingkan dengan fuzzy rule, hal ini terdapat dalam proses pemilihan kondisi untuk Diskrit Hexapod setelah modul mendapat input. Jaringan saraf tiruan dapat dengan cepat mendapatkan kondisi yang akurat untuk Diskrit Hexapod dengan menggunakan fungsi pada algoritma jaringan saraf tiruan, sedangkan untuk penggunaan fuzzy rule diperlukan suatu pengecekan pada seluruh data tabel untuk memperoleh state yang paling akurat. Fuzzy rule akan menjadi lebih lambat untuk mendapatkan kondisi yang akurat seiring dengan banyaknya jumlah data pada data tabel. Teori ini yang menjadi acuan dilakukannya penelitian lebih lanjut dengan penggunaan metode Jaringan saraf tiruan.

Elman Recurrent Neural Network merupakan salah satu metode jaringan saraf tiruan yang memiliki struktur terdiri dari input, hidden, konteks, dan output layer[4]. Metode ini dipilih karena Elman Recurrent Neural Network memiliki konteks layer akan digunakan untuk menyimpan data dari hidden layer yang nantinya akan digunakan bersama dengan input dalam penghitungan fungsi training untuk mempercepat hasil perolehan data hasil training yang akurat [2].

Sebagai studi kasus untuk modul Diskrit Hexapod ini akan dilakukan pembuatan *driving simulation game*. Dalam *driving simulation game* ini nanti akan dilakukan berbagai tes seperti gerakan melaju, melambat, membelok, menanjak, dan menurun yang nanti akan dibandingkan dengan respon oleh gerakan Diskrit Hexapod. Maka dalam skripsi ini akan dibuat modul yang dapat

mengontrol gerakan Diskrit Hexapod dengan menggunakan data hasil training yang didapat dengan menggunakan algoritma Elman Recurrent Neural Network.

2. DASAR TEORI

Diskrit Hexapod merupakan suatu *Discrete State Manipulator* yang memiliki 6 valve yang actuatornya dapat berubah menjadi antara tiga status. Valve pada Diskrit Hexapod ini dapat memiliki status memanjang, memendek, dan netral. Desain platform juga memiliki tujuan untuk mengurangi manipulator sensor juga untuk mengurangi waktu dan kompleksitas pada tampilan komputer seperti yang dijelaskan pada [6].

Elman Neural Network merupakan salah satu metode pada jaringan saraf tiruan yang sistemnya memiliki input layer, hidden layer, dan output layer di mana semuanya saling terkoneksi, dengan feedforward dan hidden layer tidak hanya terkoneksi dengan output layer tetapi juga dengan 1:1 koneksi ke layer yang lebih jauh disebut context layer [4]. Context layer merupakan layer yang berguna untuk menyimpan nilai sebelumnya dari hidden layer sehingga jaringan dapat menghasilkan suatu kondisi untuk melakukan prediksi urutan yang tidak bisa dilakukan oleh standard multilayer perceptron. [2]

Game adalah sebuah aktivitas bermain yang dilakukan dalam konteks realitas semu (pretended reality), di mana peserta permainan mencoba untuk mencapai setidaknya satu, tujuan yang tidak mudah (nontrivial) dengan cara beraksi sesuai dengan aturan yang ditentukan pada permainan tersebut [1].

Game juga efektif untuk digunakan dalam membantu untuk menghasilkan efek positif bagi pemain apabila digunakan sebagai media edukasi [8].

Simulasi permainan adalah salah satu *genre game* yang dapat membuat *gamer* mengalami situasi seperti dunia nyata pada daerah yang aman, praktis dan terkontrol [3]. Simulasi permainan juga dapat berupa simulasi kendaraan, simulasi pekerjaan, simulasi aktivitas lainnya. Simulasi kendaraan terdiri beberapa latar, dan game mekanik. Simulasi kendaraan dapat berupa kendaraan di air, di darat, dan di udara, simulasi kendaraan juga dapat tentang eksplorasi, pertandingan, atau hanya berupa pengalaman mengendarai kendaraan. [7].

Pada pengoperasiannya Elman Neural Network dapat memberikan satu atau beberapa input vektor yang dapat digunakan untuk mencari *weights* yang ideal dengan membandingkan hasil output yang didapat dalam operasi dengan hasil output yang diharapkan dengan memanfaatkan aturan pada *backpropagation* untuk mencari *delta weights* [1].

Backpropagation merupakan suatu algoritma pada jaringan saraf tiruan yang memungkinkan untuk melakukan *training* dengan suatu data set berupa *input* yang memiliki *output* yang ideal. *Backpropagation* merupakan sebuah cabang dari *feedforward* yang memiliki struktur operasi yang dapat bergerak mundur atau *backward* dan oleh karena itu disebut sebagai *backpropagation.* Backpropagation menggunakan metode optimisasi yang disebut *gradient descent* yang berguna untuk mencari *error* terkecil dari *weight*s, algoritma ini secara bertahap mengubah dan mengkalkulasi *weight* agar *output* yang dihasilkan dapat lebih mendekati *output* yang diharapkan.

Fungsi yang digunakan untuk menghitung nilai neuron adalah dengan menggunakan persamaan berikut:

$$Y = \frac{1}{1+e^{-s}} \quad (1)$$

Dengan nilai S dicari dengan persamaan berikut:

$$s = \sum_{k}^{i=1} Xi \; x \; Wi \quad (2)$$

Keterangan:

Y : Nilai *Neuron*

e : Konstanta eksponen dengan nilai 2.71828

i : bernilai 1 sampai jumlah neuron

Xi : Nilai *input* dengan *input i*

k : Banyak input dari suatu *neuron*

Wi : *Weight* dengan *input i*

Nilai *Weight* pada awalnya diinisialisasikan secara acak dengan *range* sesuai dengan persamaan berikut:

$$\frac{-2,4}{k} + \frac{2,4}{k} \quad (3)$$

Keterangan:

k : Banyak *input* dari suatu *neuron*

Proses untuk *training weight* dimulai dengan mencari nilai delta untuk layer output. Persamaan untuk mencari delta dapat dilihat sebagai berikut:

$$\delta n(p) = yn(p) \text{ x } [1 - yn(p)] \text{ x } en \quad (4)$$

Dengan nilai e didapat dengan persamaan berikut:

$$en\,(p) = yd(p) - yn(p) \quad (5)$$

Keterangan:

e_n : Nilai *Error* untuk output neuron *n*

y_d : Nilai *output* yang diharapkan dari dataset

y_n : Nilai *output neuron*

p : Nilai iterasi

δ_n : Nilai *delta* untuk output neuron *n*

Setelah nilai *error* dan *delta* untuk *output layer* dicari maka dapat dilanjutkan dengan mencari selisih nilai *weight* dengan persamaan berikut:

$$\Delta Wmn(p) = \alpha \text{ x } ym(p) \text{ x } \delta n(p) \quad (6)$$

Nilai weight untuk tiap weight yang berhubungan dengan output neuron dapat diubah setelah selisih nilai weight ditemukan dengan persamaan berikut:

$$Wmn(p + 1) = wmn(p) + \Delta wmn(p) \quad (7)$$

Keterangan:

y_m : Nilai *Hidden* neuron

ΔW_{mn} : Nilai perubahan weight yang berasal dari hidden neuron *m* menuju ouput neuron *n*

α : Nilai alpha atau *learning rate*

p : Nilai iterasi

δ_n : Nilai *delta* untuk output neuron *n*Dimana:

Setelah bagian *output* layer telah diperbarui maka akan dicari nilai delta, selisih weight, dan juga *weight* yang baru untuk *hidden layer*. Persamaan dapat dilihat berikut:

$$\delta m(p) = ym(p) \text{ x } [1 - ym(p)] \text{ x } \sum_{k=1}^{1} \delta output(p) \text{ x } w_{jk}(p) \quad (8)$$

$$\Delta Wjm(p) = \alpha \text{ x } Xj(p) \text{ x } \delta m(p) \quad (9)$$

$$Wjm(p + 1) = Wjm(p) + \Delta Wjm(p) \quad (10)$$

Keterangan:

X_j : Nilai *Input j*

y_m : Nilai *Hidden neuron*

α : Nilai *alpha* atau *learning rate*

δ_m : Nilai *delta* untuk *hidden neuron m*

p : Nilai iterasi

ΔW_{jm} : Nilai perubahan *weight* yang berasal dari *input neuron j* menuju *hidden neuron m*

W_{jm} : Nilai *weight* dari *input j* menuju *hidden neuron m*

Setelah dilakukan perhitungan untuk mengganti nilai weight untuk iterasi *p*, proses untuk training akan diulang dengan mencari nilai output, perubahan weight, hingga proses untuk update weight pada iterasi *p* + 1. Proses pengulangan akan dilanjutkan hingga akhir dari pasangan input-output pada dataset yang digunakan sebagai basis training nilai weight.

Setelah proses training nilai weight dilakukan untuk seluruh pasangan input-output pada dataset, proses dapat dilanjutkan dengan menghitung nilai SSE, SSE adalah jumlah dari tiap nilai error pada output neuron yang sebelumnya dikuadrat. Tujuan mencari nilai SSE ini adalah untuk menghitung nilai error pada proses training weight untuk 1 kali proses dataset. Nilai SSE yang besar berarti nilai weight yang dimiliki oleh sistem masih belum bisa digunakan dan harus dilakukan penghitungan ulang terhadap dataset hingga mencapai standard tertentu. Persamaan untuk menghitung nilai sum squared error (*SSE*) dapat dilihat sebagai berikut:

$$SSE = \sum_{i=1}^{n} e_i^2 \quad (11)$$

Keterangan:

e_i : Nilai *error* untuk output neuron i pada dataset

n : Banyak data pada dataset

Untuk pengujian selain dengan menggunakan SSE juga akan dilakukan penghitungan untuk RMSE atau *root-mean-square error*. Persamaan untuk RMSE adalah sebagai berikut:

$$RMSE = \sqrt[2]{\sum_{k=1}^{n} \frac{(y_k - y_k')^2}{n}} \quad (12)$$

Keterangan:

k : nilai 1 sampai jumlah data pada dataset

n : Jumlah data pada dataset

y_k : Nilai Output pada dataset

y_k' : Nilai Output hasil perhitungan

3. DESAIN SISTEM

3.1. Analisa Penggunaan Elman Neural Network

Jaringan saraf tiruan merupakan suatu metode untuk memproses informasi yang terinspirasi oleh bagaimana cara jaringan saraf manusia bekerja. Jaringan saraf tiruan memiliki kemampuan untuk menghasilkan suatu hasil keputusan dari pola data yang diberikan. Jaringan saraf tiruan memiliki banyak jenis struktur yang salah satunya adalah Elman Neural Network. Elman Neural Network memiliki struktur yang unik yaitu dengan memiliki *context layer* selain *input*, *hidden*, dan *output layer*. Elman Neural Network banyak digunakan sebagai pengenalan pola karena memiliki *context layer* yang berguna sebagai *memory*.

Elman Neural Network menggunakan operasi backpropagation sebagai *training* algoritmanya. Backpropagation memiliki kualitas untuk menghasilkan error yang minimum dengan menggunakan metode *gradient descent*. Dengan kombinasi dari backpropagation dan *context layer* yang terdapat dalam struktur Elman Neural Network makan sistem operasi dari jaringan saraf tiruan tidak hanya mendapat input dari luar atau input baru, tetapi juga mendapat input dari hasil operasi sebelumnya yang telah masuk kedalam *context layer.*

Dengan berdasarkan teori ini akan dilakukan penelitian dengan menggunakan Elman Neural Network sebagai penelitian lanjutan yang telah dilakukan sebelumnya.

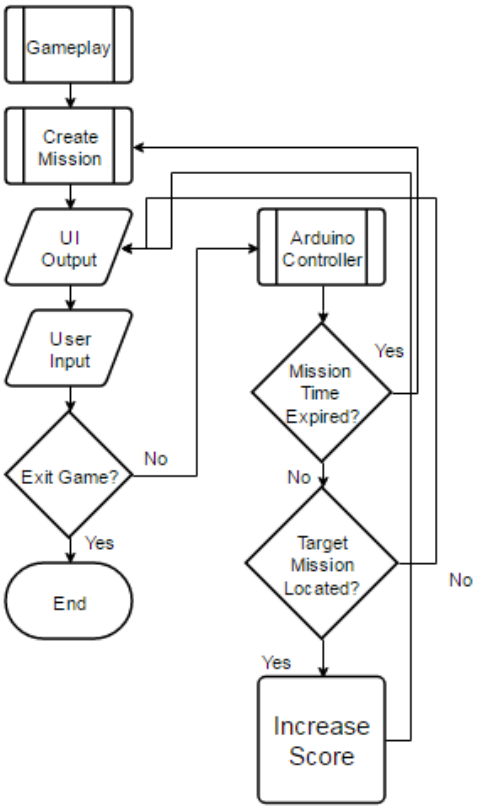
3.2. Desain Game Simulasi

Desain game dimulai dengan game menampilkan menu pilihan utama setelah program dijalankan. Pemain dapat memilih menu pilihan dengan menggerakkan kursor mouse dan memilih pilihan yang tersedia. Setelah menu pilihan dipilih maka game akan memberikan respon sesuai dengan sub-proses yang dipilih pemain..

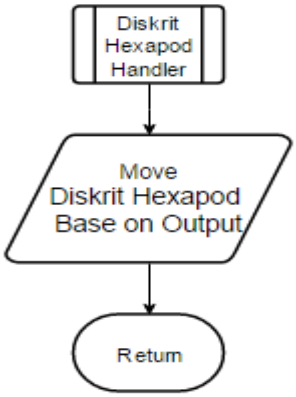
3.3 *Flowchart*

Flowchart ini menjelaskan garis besar aplikasi yang akan dibuat. Aplikasi ini dapat digunakan oleh pemain. Pemain dapat akan mendapatkan tampilan berupa menu utama dan pemain dapat melakukan pilihan untuk memulai permainan, membuka menu *option*, atau mengakhiri aplikasi. Setelah pemain memilih untuk mennjalankan permainan aplikasi akan memulai sub-proses gameplay dimana pemain dapat mengontrol jalannya obyek karakter dalam permainan. Pemain akan mendapatkan respon berupa pergerakkan Diskrit Hexapod apabila karakter dalam permainan berada pada kondisi tertentu. Pemain dapat menjalankan misi yang diberikan untuk mendapatkan *score* yang nanti akan ditampilkan pada menu *highscore*. Pemain dapat memilih menu *option* pada menu utama dan membuka tampilan yang berisi pilihan untuk mengatur hidup atau matinya suara permainan, menu pilihan untuk melihat *highscore*, dan menu pilihan untuk melihat kontrol *game*. *Flowchart* garis besar aplikasi untuk *gameplay* dapat dilihat

pada Gambar 1. *Flowchart* garis besar aplikasi untuk mengatur pergerakkan Diskrit Hexapod dapat dilihat pada Gambar 2.

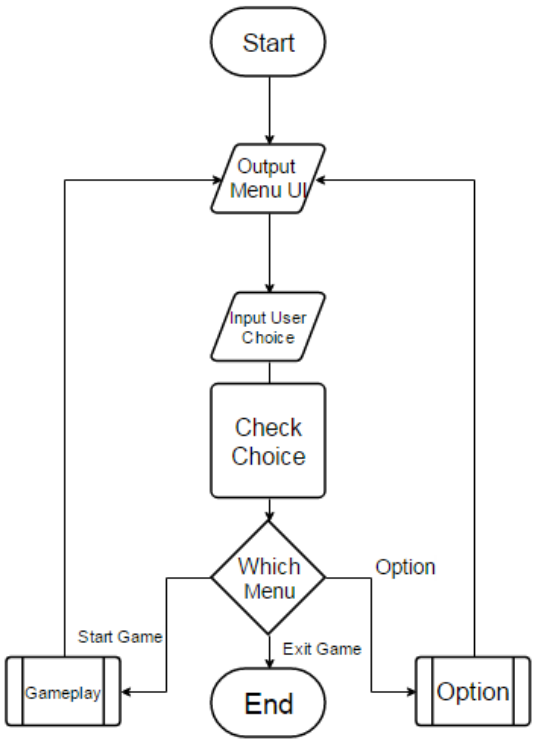


Gambar 1. *Flowchart* Garis Besar *gameplay*



Gambar 2. *Flowchart* Garis Besar Aplikasi Pergerakkan Diskrit Hexapod

Pada menu utama *game*, terdapat beberapa macam menu yang dapat dipilih. Flowchart dari menu-menu tersebut dapat dilihat pada Gambar 3.

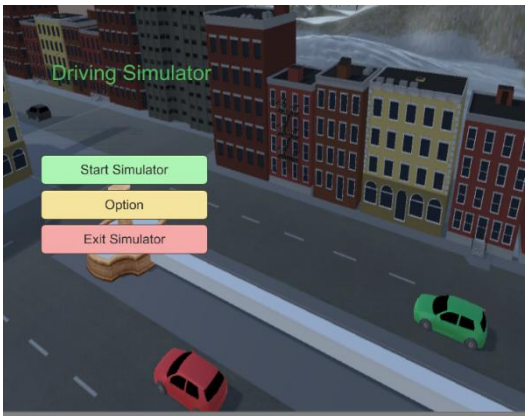


Gambar 3. Flowchart General Sistem

4. PENGUJIAN SISTEM

4.1. Pengujian Aplikasi Game Simulasi

Pada awal aplikasi untuk pengguna dijalankan akan ditampilkan tampilan menu utama dan pemain dapat melakukan aksi untuk memilih seperti pada Gambar 4.



Gambar 4. Menu Utama

Pada tampilan menu utama ini, pemain dapat melakukan pemilihan dari 3 opsi yang diberikan, apabila pemain memilih menu option akan tampil tampilan seperti pada Gambar 5.

Gambar 5. Tampilan menu Option

4.2. Pengujian Aplikasi Training

Selain game, pengujian juga dilakukan pada saat proses training weight. Salah satunya adalah dengan perngujian SSE dan RMSE. Proses perbandingan ini akan dilakukan dengan membandingkan nilai SSE dan RMSE antara 3 dataset dengan jumlah data yang berbeda yaitu dataset dengan 75 jumlah data, dataset dengan 115 jumlah data, dan dataset dengan 161 jumlah data. Pengujian akan dilakukan dengan melakukan proses pelatihan dataset dengan menghitung nilai SSE dan RMSE 3 dataset setelah dilakukan iterasi pelatihan sebanyak 1000 iterasi, 5000 iterasi, dan 10000 iterasi. Hasil proses uji coba dapat dilihat pada Tabel 1, Tabel 2, dan Tabel 3.

Tabel 1. Tabel Hasil Pengujian Nilai SSE dan RMSE untuk Dataset 75.

Output 75 Dataset	SSE 1000 Iterasi	SSE 5000 Iterasi	SSE 10000 Iterasi	RMSE 1000 Iterasi	RMSE 5000 Iterasi	RMSE 10000 Iterasi
1	10,03	0,72	0,14	0,63	0,17	0,07
2	10,01	2,11	1,27	0,63	0,29	0,22
3	6,35	3,99	3,66	0,50	0,39	0,38
4	11,44	0,37	0,10	0,67	0,12	0,06
5	11,94	4,84	0,07	0,69	0,44	0,05
6	8,68	3,62	3,72	0,58	0,38	0,38
Rata-rata	9,74	2,61	1,49	0,62	0,30	0,19

Tabel 2. Tabel Hasil Pengujian Nilai SSE dan RMSE untuk Dataset 115.

Output 115 Dataset	SSE 1000 Iterasi	SSE 5000 Iterasi	SSE 10000 Iterasi	RMSE 1000 Iterasi	RMSE 5000 Iterasi	RMSE 10000 Iterasi
1	21,89	0,65	3,19	0,93	0,16	0,35
2	22,08	1,94	3,98	0,93	0,27	0,39
3	22,17	20,6	18,6	0,94	0,90	0,86
4	14,74	22,57	0,07	0,76	0,95	0,02
5	13,65	3,39	1,78	0,73	0,64	0,26
6	9,72	4,87	0,91	0,62	0,68	0,191
Rata-rata	17,38	9,02	4,77	0,82	0,60	0,35

Tabel 3. Tabel Hasil Pengujian Nilai SSE dan RMSE untuk Dataset 161.

Output 161 Dataset	SSE 1000 Iterasi	SSE 5000 Iterasi	SSE 10000 Iterasi	RMSE 1000 Iterasi	RMSE 5000 Iterasi	RMSE 10000 Iterasi
1	32,08	28,47	26,88	1,13	1,06	1,03
2	30,69	5,00	26,41	1,10	0,44	1,02
3	4,34	3,42	3,321	0,41	0,37	0,36
4	27,04	17,56	2,70	1,04	0,83	0,32
5	27,79	2,55	1,49	1,05	0,31	0,24
6	4,871	3,12	3,58	0,44	0,35	0,37
Rata-rata	21,13	10,02	10,73	0,86	0,56	0,56

4.3. Penghitungan Nilai Waktu Aktivasi

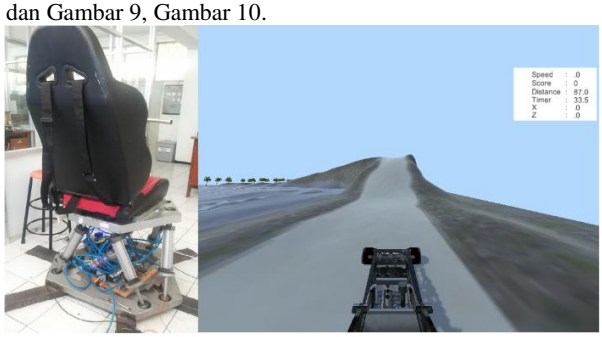
Penghitungan waktu aktivasi dilakukan dengan membandingkan waktu yang dibutuhkan oleh aplikasi Flight Simulator dan aplikasi Driving Simulator. Penghitungan waktu aktivasi ini dilakukan untuk melihat apakah jumlah data mempengaruhi waktu untuk aktivasi dan seberapa banyak selisih waktu yang dihasilkan dengan jumlah data yang berbeda. Hasil pengukuran proses penghitungan waktu aktivasi dapat dilihat pada Tabel 4.

Tabel 4. Tabel Perbandingan Waktu Aktivasi

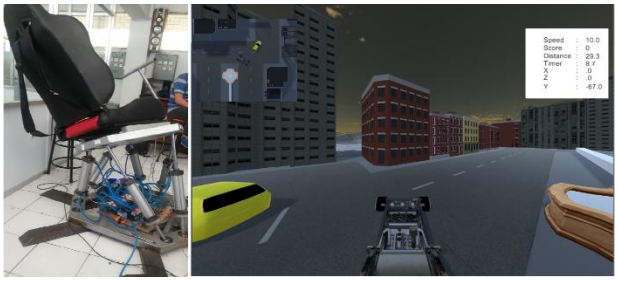
Jumlah Data	Waktu Flight Simulator (Arduino)	Waktu Driving Simulator (Arduino)
9	2.3ms	8ms
27	3.2ms	9ms
60	4ms	9ms
75	4.8ms	8ms
115	6.8ms	9ms

4.4. Pengujian Pergerakkan Diskrit Hexapod

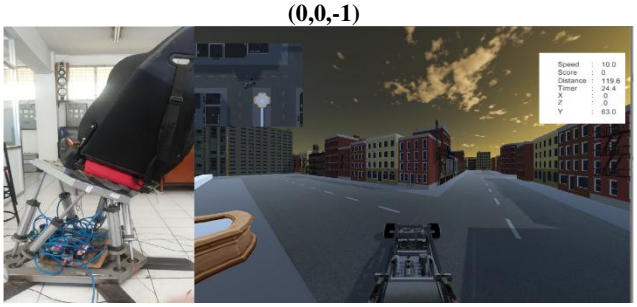
Pada pengujian pergerakkan Diskrit Hexapod dilakukan pengambilan beberapa gambar dengan input yang berbeda. Contoh beberapa hasil tes dapat dilihat pada Gambar 6, Gambar 7, Gambar 8, dan Gambar 9, Gambar 10.



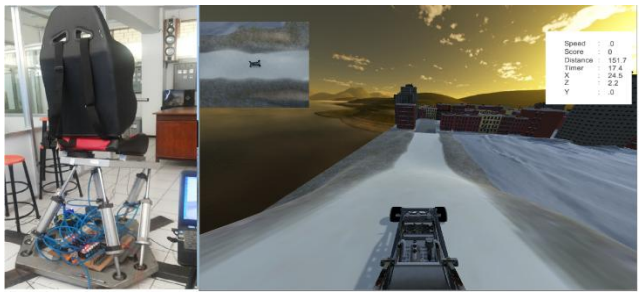
Gambar 6. Gerakkan Diskrit Hexapod dengan dengan input (0,0,0)



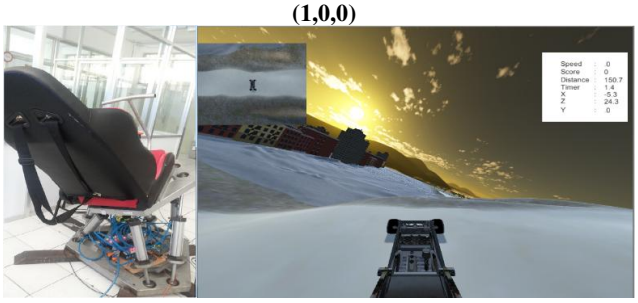
Gambar 7. Gerakkan Diskrit Hexapod dengan dengan input (0,0,-1)



Gambar 8. Gerakkan Diskrit Hexapod dengan dengan input (0,0,1)



Gambar 9. Gerakkan Diskrit Hexapod dengan dengan input (1,0,0)



Gambar 10. Gerakkan Diskrit Hexapod dengan dengan input (0,1,0)

5. Kesimpulan

Berdasarkan hasil pengujian dapat disimpulkan beberapa hal sebagai berikut:

- Nilai SSE yang didapat setelah dilakukan proses pelatihan sebanyak 1000, 5000, dan 10000 dapat dilihat mengalami proses pengurangan yang dapat dilihat dari Tabel 5.1, Tabel 5.2, dan Tabel 5.3 yang menunjukkan bahwa error pada output semakin kecil
- Hasil dari perbandingan waktu proses aktivasi metode Elman Neural Network dengan metode fuzzy rule dapat digunakan untuk menarik kesimpulan bahwa waktu yang digunakan untuk proses aktivasi dengan menggunakan metode Elman Recurrent Neural Network dengan menggunakan dataset dengan jumlah data hingga 115 masih lebih besar daripada metode terdahulu
- Nilai RMSE yang didapat untuk tiap output neuron untuk target SSE 0,1 berkisar pada nilai 0,0632.
- Jumlah Dataset yang digunakan untuk melakukan *training* nilai *weight* adalah 75, 115, dan 161 pasangan *input* berupa sudut x, sudut z, dan sudut y.

6. DAFTAR PUSTAKA

[1] Adams, E. 2010. Fundamentals of *Game* Design. Pearson Education, Inc.

[2] Bi Yaxin, Williams Mary Anne. Knowledge Science, Engineering and Management. 4th International Conference 2010, 64-66. Springer

[3] Bown, Johnathan. 2016. Emotions, Technology, and Digital Games. Elsevier Science & Technology Books

[4] Cruse, Holk 2006. Neural Networks as Cybernetic Systems – 2nd and revised edition. Brains, Minds & Media

[5] Hagan, M.T. B, Howard. H, Mark. 2014. Neural Networks Design 2nd Edition. Martin T Hagan

[6] Pasila F. and Alimin R. 2013. Designing the 6-DOF Massive Parallel Arrays with Artificial Intelligence Control, Proceedings of the 2nd International Conference on International Conference on Robotic Automation System (ICORAS 2013).

[7] Pasila, F. Budhi, Gregorius S. Sutantra, Andreas. Juwiantho, Hans. 2014. Cockpit View Vehicle *Game* Simulators For Massive Parallel Arrays Platform Based On Neural-Fuzzy Systems. ARPN Journal of Engineering and Applied Sciences, VOL. 9, NO. 12.

[8] Prot, S., Anderson, C. A., Gentile, D. A., Brown, S. C., & Swing, E. L. 2014. In A. Jordan & D. Romer (Eds). The Positive and Negative Effect of Video Game Play. Media and the Well-Being of Children and Adolescents (109–128). New York : Oxford University Press.