Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 11, November 2019, hlm. 10667-10677 http://j-ptiik.ub.ac.id

Penerapan Jaringan Saraf Tiruan Untuk Bantuan Pengereman Pemain Pada Gim Balapan

Mochamad Halim[1], Muhammad Aminul Akbar[2], Tri Afirianto[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]m.halim180@gmail.com, [2]muhammad.aminul@ub.ac.id, [3]tri.afirianto@ub.ac.id

Abstrak

Gim balapan adalah genre gim elektronik bertema kompetisi balap yang mana pemain berkompetisi melawan pemain lain atau komputer untuk mencapai garis akhir dengan waktu penyelesaian lintasan tercepat. Gim balapan berfokus pada kemampuan refleks, teknik balapan dan pengetahuan lintasan, sehingga wajar bila pemain pemula banyak melakukan kesalahan seperti keluar lintasan atau menabrak dinding. Salah satu teknik yang sulit dikuasai pemain pemula adalah pengaturan waktu rem. Kesalahan dalam pengaturan waktu rem dapat menyebabkan pemain menabrak pembatas lintasan, menambah waktu yang dibutuhkan untuk menyelesaikan lintasan. Untuk meningkatkan performa pemain pemula, diperlukan bantuan pengereman untuk membiasakan dan membantu pemain pemula dengan menirukan pengaturan waktu rem seorang pemain ahli. JST (Jaringan Saraf Tiruan) adalah algoritme yang dapat mempelajari pola dari sekumpulan data dan memiliki kemampuan untuk mengurangi kesalahan, sehingga algoritme ini cocok untuk diterapkan dalam bantuan pengereman. Metodologi yang digunakan untuk mengembangkan sistem bantuan pengereman adalah metode JST. Sistem bantuan pengereman diimplementasikan dalam *framework racing game starter kit*, sebuah paket asset dari *Unity Asset Store*. Data balapan pemain ahli direkam, lalu sebuah algoritme dilatih mempelajari data tersebut. Semua kebutuhan fungsional JST telah divalidasi menggunakan pengujian *blackbox*, pengujian *F1 Score* 0,78 dan pengujian penerimaan pengguna menunjukkan penurunan jumlah tabrakan, tapi waktu penyelesaian lintasan bertambah.

Kata kunci: *bantuan rem, gim balapan, jaringan saraf tiruan*

Abstract

Racing game is a video game genre themed around racing event, in which a player compete againts another player or computer to reach the finish line with the shortest clear time. A racing game focused around player reflex, racing technique and track knowledge. It is common for a new player to make many errors, for example driving outside the lane and crashing into the road barrier. For a new player, brake timing is one of the hardest skill to master. Wrong brake timing can cause player to hit the road barrier, or add more to the clear time. A brake assistance is needed to improve a player racing performance by imitating an advanced player brake timing. An advanced player's racing data is recorded, then it is used as training data by an algoritm. ANN (Artificial Neural Network) is an algoritm that can learn the pattern of a data set and have the ability to reduce error, thus this algoritm is acceptable for the brake assist. The metodology used to develop this brake assist is the ANN method. The brake assist system is implemented in the framework of racing game starter kit, an asset package from Unity Asset Store. ANN testing by using blackbox testing validate functionality needs, F1 testing results 0.78 and user acceptance testing shows lower crash count, but increase in total lap time.

Keywords: *brake assist, racing game, artificial neural network*

1. PENDAHULUAN

Genre gim balapan mencakup semua gim yang mana pemain mengendarai mobil atau kendaraan lainnya dalam sebuah lintasan dengan tujuan mencapai waktu balap tercepat (Gregory, 2009). Unsur utama gim balapan berfokus pada keputusan cepat, refleks dan pengetahuan tentang lintasan. Menurut Akito Sugawara dalam *Gran Turismo 5 Official Strategy Guide* (2010), dibutuhkan waktu untuk menguasai suatu lintasan balapan. Dalam proses mempelajari lintasan ini, salah satu tantangan yang sulit dihadapi seorang pemain pemula adalah pengaturan waktu mengerem. Mengerem adalah teknik dasar dalam balapan yang mana pemain mengurangi kecepatan mobil pada waktu yang tepat agar mobil lebih mudah berbelok. Kesalahan dalam pengaturan waktu rem dapat menyebabkan mobil pemain keluar lintasan atau dibalap oleh pemain lain, oleh karena itu diperlukan implementasi bantuan pengereman untuk pemain pemula.

JST (*Jaringan Saraf Tiruan*) adalah algoritme yang meniru cara kerja otak (Heyakin, 2009). Algoritme JST dapat digunakan sebagai algoritme bantuan pengereman karena memiliki kemampuan mempelajari sebuah *dataset*, mengurangi *error* dan adaptasi yang tinggi, sehingga cocok untuk diterapkan dalam bantuan pengereman. JST harus dilatih terlebih dahulu sebelum diimplementasi. Pertama data balap pemain ahli harus dikumpulkan. Pelatihan dilakukan dengan pemberian sebagian data balap pemain ahli sebagai data latih JST, sementara sebagian data balap lainnya akan digunakan sebagai data uji untuk menguji ketepatan keluaran JST.

Racing game starter kit adalah sebuah paket *asset* dari *Unity Asset Store* yang dibuat oleh *Ian_GameDev*, paket *asset* ini akan digunakan sebagai *framework* dasar untuk membangun sistem bantuan pengereman. *Racing game starter kit* telah menyediakan model 3D, lintasan, serta *AI* untuk lawan balapan, akan tertapi masih belum ada fitur bantuan pengereman dalam *racing game starter kit*, sehingga dapat diimplementasikan bantuan pengereman dalam sistem kontrol mobil pemain. Implementasi sistem bantuan pengereman ini diharapkan dapat meningkatkan performa pemain.

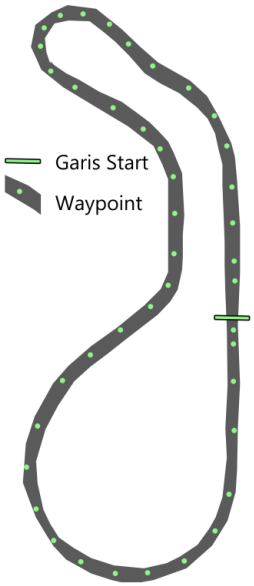
2. LANDASAN KEPUSTAKAAN

2.1 Racing Game Starter Kit

Menurut halaman *Unity Asset Store*, *Racing Game Starter Kit* adalah aset siap pakai yang dapat digunakan pada *editor unity* untuk membuat gim balapan dengan mudah. Gambar 1 adalah tampilan *racing game starter kit* saat dimainkan. *Racing game starter kit* telah menyediakan model 3D, fisika mobil dan sepeda motor, lintasan, berbagai jenis mode balapan, serta AI untuk lawan balap. Halaman unity asset store racing game starter kit dapat diakses pada link, https://assetstore.unity.com/packages/templates/racing-game-starter-kit-22615.

Gambar 1. Screenshot racing game starter kit

Waypoint adalah sekumpulan *Game Object* yang berupa titik-titik dalam lintasan *racing game starter kit* untuk menghitung proses penyelesaian lintasan pemain. *Waypoint* juga dapat digunakan untuk merekam kondisi di sekitar mobil pemain pada waktu pencatatan data. Gambar 2 menampilkan lokasi *waypoint*.

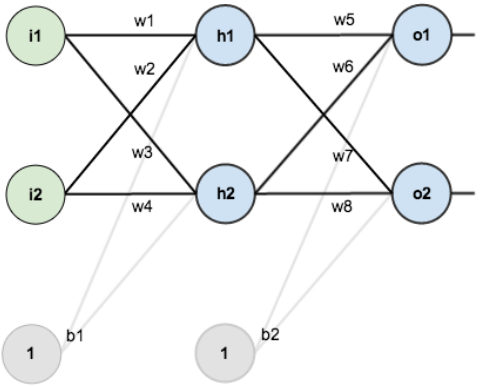


Gambar 2. Waypoint terletak di tengah garis-garis hijau

2.2 Jaringan Saraf Tiruan

Otak adalah komputer yang sangat kompleks, non-linear dan pararel, serta dapat mengatur unsur strukturalnya untuk melakukan kalkulasi yang jauh lebih cepat dari komputer digital (Heykin, 2009). JST (*Jaringan Saraf Tiruan*) adalah algoritme yang menirukan cara kerja otak untuk melakukan sebuah pekerjaan tertentu.

Seperti yang digambarkan pada Gambar 3, struktur sebuah JST umumnya memiliki tiga bagian, *input layer*, *hidden layer* dan *output layer*. Sebuah *layer* adalah satu atau lebih neuron yang dikelompokkan. *Input layer* adalah *layer* yang *neuron*nya menerima variabel masukan dari luar sistem, jumlah *neuron input layer* sama dengan jumlah masukan dari luar sistem. *Hidden layer* adalah *layer* yang melakukan kalkulasi berdasarkan masukan dari *input layer* atau *hidden layer* lainnya dan meneruskan hasil variabel keluarannya ke *hidden layer* berikutnya atau *output layer*, jumlah *hidden layer* disarankan tidak terlalu jauh dari jumlah variabel masukan dari luar sistem. *Output layer* adalah *layer* akhir yang menghasilkan variabel keluaran akhir dari sistem.



$i = inputNeuron$ $b = bias$
$h = hiddenLayer$ $w = weight$
$o = outputLayer$

Gambar 3. Diagram struktur JST

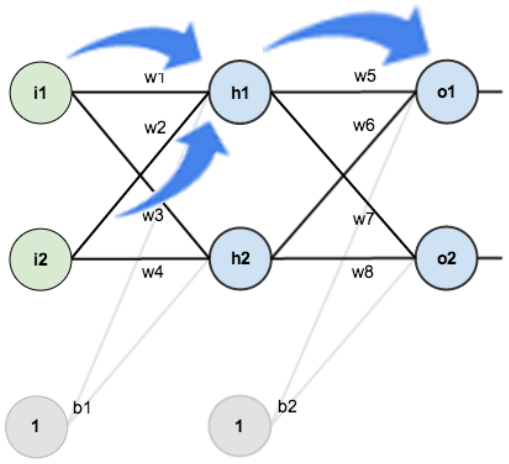
Weight adalah nilai yang diberikan *neuron* terhadap tiap masukannya, semakin besar nilai *weight* maka semakin besar pengaruh sebuah masukan terhadap keluaran sebuah *neuron*. *Bias* adalah nilai tambahan sebuah *layer* yang mempengaruhi keluaran *neuron-neuron*nya, tujuan *bias* adalah mencegah *neuron* mengeluarkan hasil nol yang dapat menggangu kestabilan kalkulasi. Fungsi aktifasi adalah fungsi yang mengubah mengubah hasil keluaran ke dalam jarak keluaran yang diharapkan.

Sebuah *neuron* menerima masukan dari luar sistem atau *layer* sebelumnya. Kemudian *neuron* menghitung nilai *output*, hasil keluaran *neuron* seperti pada Persamaan (1). Variabel *N* adalah keluaran *neuron* sebelum diaktifasi. Variabel *input* adalah variabel masukan dari *neuron-neuron* di *layer* sebelumnya. Setelah itu hasil *output* diaktifkan dengan fungsi aktifasi pada Persamaan (2) dan diteruskan ke *layer* berikutnya. Variabel *e* adalah konstanta matematika.

$$N = \sum (weight * input) - bias \quad (1)$$

$$output = \frac{e^N}{(1+e^N)} \quad (2)$$

Untuk mendapatkan keluaran dari JST, sistem akan memberi nilai masukan pada *input layer*, kemudian masukan-masukan tersebut diproses ke *neuron-neuron* di *layer hidden* berikutnya, lalu keluaran dari *hidden layer* tersebut digunakan sebagai masukan untuk *hidden layer* atau *output layer* berikutnya, proses ini dapat berulang hingga *output layer* menghasilkan variabel keluaran, proses ini disebut penyebaran ke depan, seperti yang digambarkan pada Gambar 4.



Gambar 4. Diagram penyebaran ke depan

Keistimewaan JST adalah kemampuannya untuk mengubah *weight* dan *bias neuron-neuron*nya berdasarkan seberapa jauh kesalahan hasil keluaran sistem dari hasil keluaran yang diharapkan dari *dataset*. Sebelum diimplementasikan, JST harus dilatih terlebih dahulu. Pada tahap ini JST akan diberikan data latih sebagai *input* beserta hasil keluaran JST yang diharapkan, kemudian JST melakukan penyebaran ke depan hingga menghasilkan keluaran. Setelah hasil keluaran didapatkan, JST membandingkan hasil data latih dengan keluaran JST menggunakan Persamaan (3). Variabel *desiredOutput* adalah keluaran dari data latih dan *output* adalah keluaran dari JST. Variabel *error* adalah perbedaan antara *output* JST dengan hasil yang diharapkan.

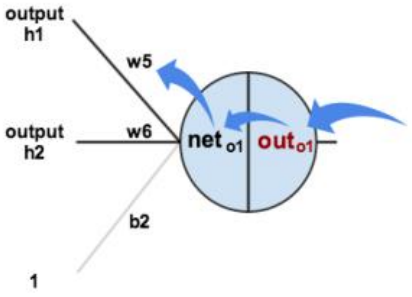
$$error = desiredOutput - output \quad (3)$$

Hasil *error* tersebut kemudian digunakan untuk menghitung *errorGradient* menggunakan Persamaan (4). Variabel *errorGradient* adalah seberapa besar kontribusi *weight* untuk menghasilkan *error* dari *output* tersebut.

$$errorGradient = output * (1 - output) * error \quad (4)$$

Setelah hasil *errorGradient* ditemukan, *weight* akan diubah berdasarkan *alpha* menggunakan Persamaan (5). Variabel *alpha* adalah kecepatan belajar JST, variabel *alpha* dapat bervariasi dari satu sistem JST ke sistem JST lain, sehingga *variabel* ini harus dicoba hingga mendapatkan *alpha* yang memberikan perubahan *weight* yang bagus, proses ini digambarkan oleh Gambar 5.

$$weight = \sum \left(\begin{matrix} alpha * input * \\ (desiredOutput - output) \end{matrix} \right) \quad (5)$$



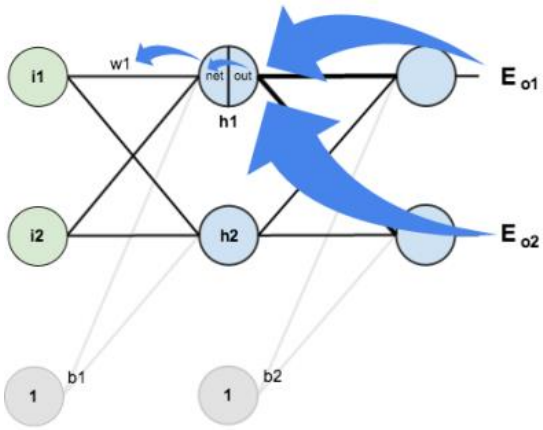
Gambar 5. Update *weight* untuk *output layer*

Setelah *errorGradient layer output* diketahui, maka sistem akan mengambil *errorGradient* dari *layer* berikutnya dan menghitung *errorGradient* untuk *neuron* tersebut menggunakan Persamaan (6). Variabel *nextLayerErrorGradient* adalah *errorGradient* dari *layer* berikutnya. Setelah nilai *errorGradient* ditemukan, *neuron* akan memperbarui nilai *weight* menggunakan Persamaan (7). Proses ini berulang mundur

hingga *hidden layer* pertama, Gambar 6 menggambarkan proses ini.

$$weight = \sum \begin{pmatrix} alpha * input \\ * errorGradi\ ent \end{pmatrix} \quad (6)$$

$$errorGradi\ ent = \begin{pmatrix} output * (1 - output) * \\ \sum (nextLayerErrorGradie\ nt) \end{pmatrix} \quad (7)$$



Gambar 6. Update *weight* dari *output layer* ke *hidden layer*

Setelah *errorGradient* pada suatu *layer* dihasilkan, maka nilai *bias* juga akan diperbarui menggunakan Persamaan 8. Pembaruan *weight* dan *bias* ini dilompati saat JST selesai dilatih.

$$bias = \sum (alpha * -1 * errorGradi\ ent) \quad (8)$$

2.3 Pengujian *F1 Score.*

Pengujian akurasi tidak selalu akurat bila data yang tersedia tidak seimbang (Powers, 2012). *Confusion matrix* atau matriks kebingungan, adalah tabel yang menggambarkan kemungkinan hasil sebuah sistem klasifikasi seperti digambarkan pada Tabel 1. Jika hasil prediksi sama dengan hasil data, maka jika positif disebut *true positive*, jika negatif disebut *true negative*.

Jika hasil prediksi positif, tapi hasil data negatif, maka disebut *false positive*. Kesalahan ini disebut *Type I error*.

Jika hasil prediksi negatif, tapi hasil data positif, keluaran maka disebut *false negative*. Kesalahan ini disebut *Type II error*.

Tabel 1. *Confusion Matrix*

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Variabel *precision* adalah seberapa besar hasil algoritme yang relevan, dibandingkan dengan seluruh hasil keluaran algoritme. Nilai ini lebih diutamakan jika kesalahan sebuah *false positive* cukup beresiko. Rumus *precision* dapat dilihat pada Persamaan (9).

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (9)$$

Variabel *recall* adalah seberapa besar hasil algoritme yang relevan dibandingkan dengan hasil benar dari seluruh data yang disajikan. Nilai ini sangatlah penting bila kesalahan sebuah *false negative* memiliki dampak buruk. Persamaan (10) menggambarkan rumus mencari nilai *recall*.

$$recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (10)$$

F1 score adalah nilai campuran dari nilai *precision* dan nilai *recall*. Nilai *F1 score* menggambarkan secara seimbang keluaran *false positive* dan *false negative* dalam sistem. Persamaan (11) digunakan untuk mencari nilai *F1 score*.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \quad (11)$$

3. METODOLOGI PENELITIAN

Penelitian ini adalah tipe implementatif pengembangan lanjut karena penelitian ini mengimplemetasikan JST pada fitur bantuan pengereman pada *framework racing game starter kit*. Gambar 7 adalah alur penelitian yang dilakukan.

Gambar 7. Diagram alur penelitian

3.1. Studi Literatur

Dalam tahap studi literatur, sumber-sumber literatur akan dikaji untuk dijadikan dasar dalam pengembangan sistem antara. Beberapa studi literatur yang dilakukan penulis adalah gim balapan, bantuan pengereman, Unity, *racing game starter kit*, JST, pengujian *blackbox*, pengujian F1 *score*, dan pengujian penerimaan pemain.

3.2. Analisis Kebutuhan

Dalam tahap analisis kebutuhan, setelah kebutuhan-kebutuhan dianalisis, kebutuhan data sistem akan dikumpulkan.

3.3. Perancangan Dan Implementasi

Pada tahap perangcangan, akan dirangkai rencana implementasi program berdasarkan kebutuhan yang didapatkan dalam tahap analisis kebutuhan. Setelah perencanaan selesai, rancangan diimplementasi menjadi program. JST menggunakan data latih yang telah dikumpulkan pada tahap analisis kebutuhan. JST yang telah dilatih akan diimplementasikan pada kode kendali mobil untuk pengujian. Iterasi terbaik hasil tahap perancangan akan diimplementasi pada *racing game starter kit* menggunakan *editor* Unity. Jika hasil performa bantuan pengereman masih menghasilkan banyak kesalahan, maka data harus ditambah atau diganti dan JST harus dilatih kembali.

3.4. Pengujian

Pada tahap ini, dilakukan tiga jenis pengujian terhadap program yang telah diimplementasi. Pengujian *blackbox* untuk validasi kebutuhan fungsional, pengujian *F1 score* untuk menguji kecenderungan keluaran sistem dan pengujian penerimaan pengguna untuk menguji kecocokan sistem dengan pemain pemula. Data yang dicatat dalam pengujian penerimaan pengguna antara lain:

1. Waktu balap pemain tiap putaran, dibandingkan sebelum dan sesudah menggunakan bantuan pengereman.
2. Jumlah tabrakan mobil pemain dengan pembatas lintasan.
3. Perbandingan *frame per second* sebelum dan sesudah menggunakan bantuan pengereman.

3.5. Kesimpulan Dan Saran

Kesimpulan ditarik setelah mengolah hasil pengujian. Kesimpulan diperlukan sebagai jawaban dari rumusan masalah penulisan ini. Saran diberikan oleh peneliti terhadap hasil dari masing-masing tahap penelitian, yang digunakan untuk memperbaiki kualitas penilitian, baik untuk kualitas penelitian ini sendiri, atau kualitas penelitian oleh orang lain yang berhubungan dengan penelitian ini, di masa yang akan datang.

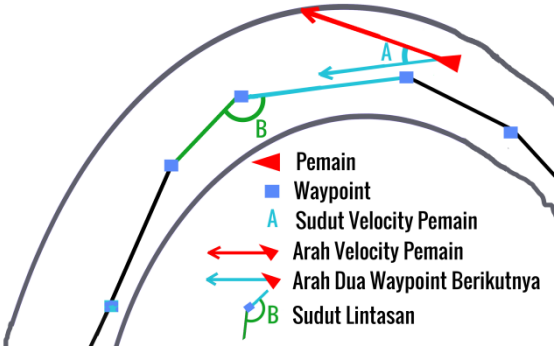
4. Analisis Kebutuhan

Analisis kebutuhan fungsional adalah pengumpulan fungsi-fungsi yang diperlukan untuk kerja sistem. Kebutuhan fungsional didapatkan dari analisa fungsi-fungsi yang kurang dalam *racing game starter kit*. Dari tahap analisa kebutuhan, ditemukan 10

kebutuhan fungsional.

Analisis kebutuhan data adalah pengumpulan data-data yang diperlukan untuk tahap pelatihan dan pengujian JST. Data balapan dikumpulkan dari seorang pemain ahli. Seorang pemain akan dikategorikan sebagai pemain ahli bila bisa yang menyelesaikan tiga putaran lintasan tanpa keluar lintasan dengan waktu dibawah tiga menit. Gambar 8 menampilkan tiga data penting yang dikumpulkan dari perfoma pemain ahli tiap dua detik. Data-data yang dikumpulkan dari pemain ahli, antara lain:

1. Kecepatan mobil pemain. Kecepatan dalam *miles per hour* yang ditampilkan pada UI pemain.
2. *Sudut velocity*, yaitu sudut yang dibuat antara arah laju mobil dengan arah kedua waypoint di depan pemain.
3. Sudut lintasan. Sudut yang dibuat oleh tiga waypoint di depan mobil pemain.



Gambar 8. Waypoint dan data sudut yang dicatat

5. PERANCANGAN DAN IMPLEMENTASI

5.1. Perancangan Sistem

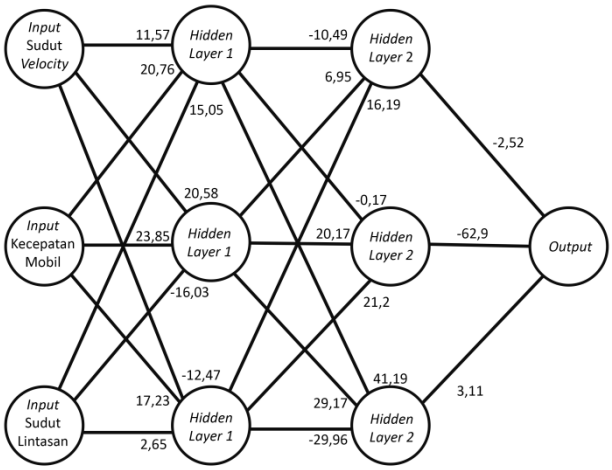
Pada tahap ini dibuat *flowchart* yang menggambarkan cara kerja sistem. Perancangan melimputi persiapan algoritme JST pada fungsi *start* dan penjalanan JST pada fungsi *update*.

Persiapan JST meliputi deklarasi, pelatihan, dan pengujian. Deklarasi akan memuat JST yang telah tersimpan atau membuat JST baru bila *file* tidak ada dalam lokasi tujuan. Dalam pelatihan, JST akan diberi 70% data pemain ahli sebagai data latih, dan pelatihan dilakukan hingga 1000 kali. JST yang telah dilatih disimpan agar dapat dilatih lebih dalam atau disimpan untuk penggunaan. Pengujian mencari nilai *F1 Score* untuk pengujian menggunakan 30% data sebagai data uji.

Perancangan penjalanan JST dalam fungsi *update*. Merancang cara JST mendapatkan variabel-variabel penting untuk masukan JST.

5.2. Implementasi Sistem

Pada tahap ini, rancangan sistem diimplementasikan ke *racing game starter kit* menggunakan *editor* Unity. Saat objek mobil dibuat, JST yang telah dilatih akan dimuat. Fungsi *brakeCheck* akan memeriksa kondisi lintasan tiap *frame* dan menjadikan masukan algoritme JST, bila hasil di atas nilai tertentu, maka hasil JST dijadikan kekuatan rem tipe *footbrake*. Program akan dieksport dari *editor* Unity menjadi *file executable* untuk mempermudah pengujian. Gambar 9 adalah struktur JST yang telah dilatih dan diimplementasikan.



Gambar 9. Struktur JST setelah pelatihan

6. PENGUJIAN

6.1. Pengujian fungsional

Pengujian pertama adalah pengujian validasi fitur program yang telah dirumuskan pada tahap analisis kebutuhan menggunakan *black box testing*. Pengujian ini dilakukan oleh seorang penguji tanpa perlu mengetahui cara kerja sistem. Dari 10 kebutuhan fungsional yang dirumuskan dalam analisis kebutuhan, semuanya tervalidasi.

6.2. Pengujian *F1 Score*

Pengujian *F1 score* dilakukan dengan menghitung nilai *precision, recall* dan *F1 score*. Gambar 10 adalah tabel kebingungan yang menggambarkan hasil keluaran JST.

Pengujian menggunakan Persamaan 9 untuk mendapat nilai *precision* 0,69. Nilai *precision* ini kurang sempurna. Sistem bantuan pengereman terkadang mengerem pada situasi yang tidak memerlukan rem, sehingga berpotensi menambah waktu yang dibutuhkan pemain untuk menyelesaikan lintasan.

Pengujian menggunakan Persamaan 10 untuk mendapat nilai *recall* 0,9. Nilai *recall* ini sangat baik. Sistem bantuan pengereman mengerem pada sebagian besar situasi yang membutuhkan rem, menyebabkan pengurangan jumlah tabrakan pemain dalam lintasan.

Jika nilai *precision* dan *recall* digabungkan menggunakan Persamaan 11, maka nilai *F1 score* adalah 0,78. Nilai *F1 score* ini menggambarkan sistem bantuan pengereman yang cukup baik saat dihadapkan dengan data uji.

		Predicted	
		Negative	Positive
Actual	Negative	24	8
	Positive	2	18

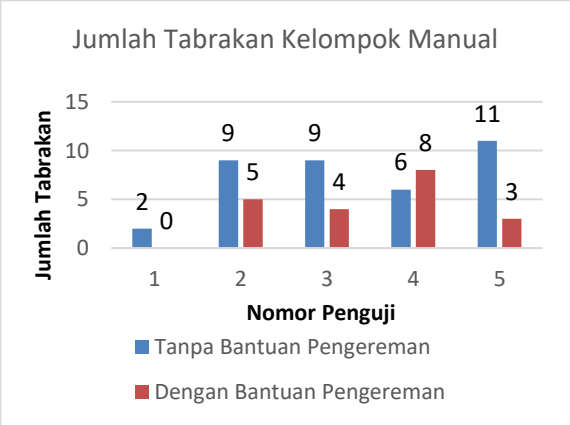
Gambar 10. Waypoint dan data sudut yang dicatat

6.3. Pengujian Penerimaan Pengguna

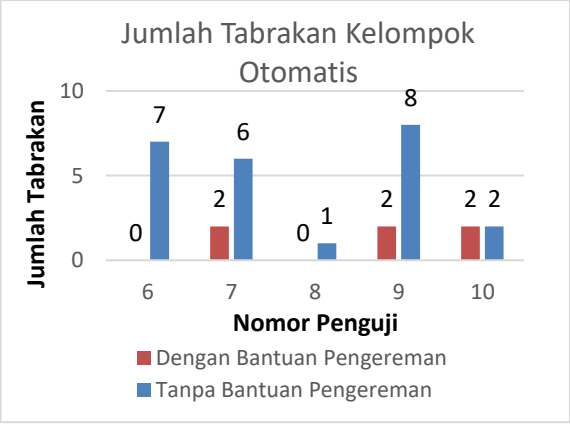
Pengujian penerimaan pengguna dilakukkan dengan pencatatan performa penguji saat sebelum dan sesudah menggunakan bantuan pengereman. Penguji adalah pemain pemula yang belum pernah mencoba lintasan yang digunakan dalam penelitian.

Pengujian akan dilakukan dua kali, penguji akan dibagi menjadi dua kelompok karena kecenderungan peningkatan performa pemain pada percobaan ke dua. Kelompok pertama adalah kelompok manual yang menguji tanpa bantuan pengereman terlebih dahulu. Kelompok kedua adalah kelompok otomatis yang menguji dengan bantuan pengereman terlebih dahulu.

Gambar 11 menggambarkan perbandingan jumlah tabrakan kelompok manual, sementara Gambar 12 adalah kelompok otomatis. Dari kedua grafik, dapat dilihat bahwa penggunaan bantuan pengereman mengurangi jumlah tabrakan pemain.



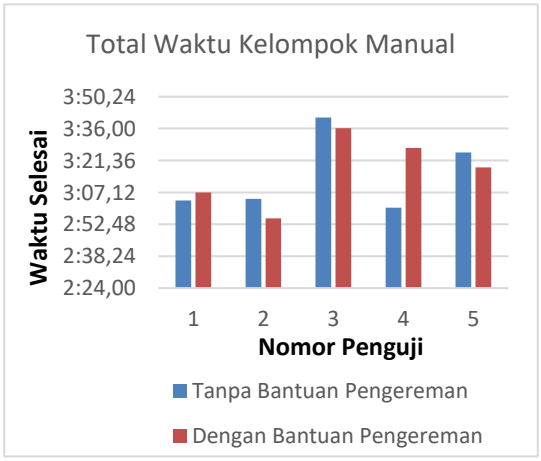
Gambar 11. Grafik jumlah tabrakan kelompok manual



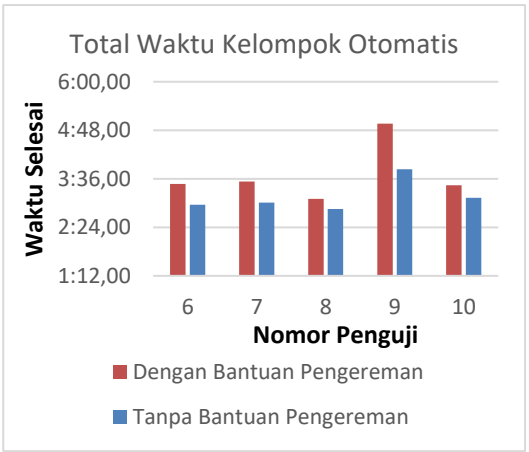
Gambar 12. Grafik jumlah tabrakan kelompok

otomatis

Gambar 13 menggambarkan perbandingan total waktu penyelesaian lintasan kelompok manual, sementara Gambar 14 untuk kelompok otomatis. Pada sebagian besar penguji kelompok manual, bantuan pengereman berhasil mengurangi waktu penyelesaian lintasan. Akan tetapi, performa seluruh kelompok otomatis lebih bagus saat tidak menggunakan bantuan pengereman. Rata-rata waktu penyelesaian kelompok manual adalah 3 menit 15 detik, lebih cepat dari pada kelompok otomatis yang rata rata waktu penyelesaiannya 3 menit 25 detik. Pada percobaan pertama, waktu penyelesaian kelompok manual lebih cepat. Pada percobaan kedua, waktu penyelesaian kelompok otomatis lebih cepat.



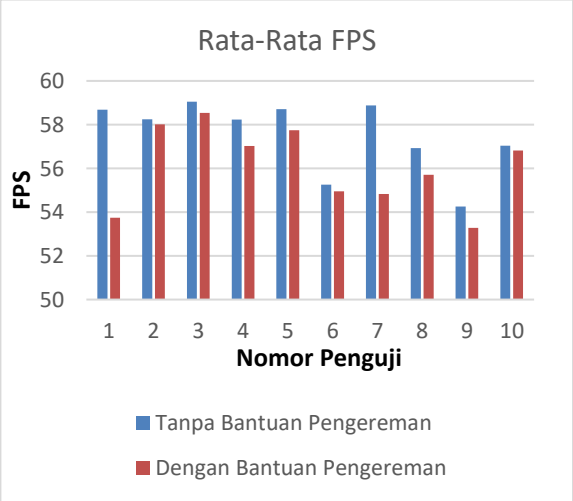
Gambar 13. Grafik waktu penyelesaian lintasan kelompok manual



Gambar 14. Grafik jumlah tabrakan kelompok manual

Berbeda dengan pengujian performa pemain, pengujian FPS tidak terpengaruh oleh pengalaman penguji. Oleh karena itu, hasil pengujian FPS tidak perlu dipisah. Gambar 15 menggambarkan rata-rata FPS saat permainan berjalan. Sembilan dari sepuluh pemain mengalami penurunan rata-rata FPS saat menggunakan bantuan pegereman. Bantuan pengereman menurunkan rata-rata FPS saat dijalankan.

Rata-rata penurunan FPS adalah 1,46.FPS tertinggi yang dicapai tanpa bantuan pengereman adalah 59,05, sementara saat menggunakan bantuan pengereman adalah 58,54. FPS terendah yang dicapai tanpa bantuan pengereman adalah 54,25, sementara saat menggunakan bantuan pengereman adalah 53,28.



Gambar 15. Grafik jumlah tabrakan kedua kelompok

7. KESIMPULAN DAN SARAN

Kebutuhan fungsional bantuan pengereman untuk *framework racing game starter kit* memerlukan penyimpan data pemain ahli, pelatih algoritme JST, penguji algoritme JST dan metode pengereman otomatis menggunakan *footbrake*. Pada pengujian validasi, semua

kebutuhan dalam analisis kebutuhan telah terpenuhi.

Implementasi bantuan pengereman menghasilkan nilai *F1 score* 0,78 yang lebih berat pada nilai *recall*. Nilai *precision* yang rendah menyebabkan banyak keluaran *false positive* yang berpotensi membuat mobil lebih sering mengerem, menaikan waktu yang dibutuhkan pemain untuk menyelesaikan lintasan. Nilai recall bagus, sehingga mobil berpotensi mengerem pada kondisi yang sesuai, sehingga jumlah tabrakan berkurang.

Pengujian performa pemain menunjukkan bahwa bantuan pengereman mengurangi jumlah tabrakan pemain dengan rata-rata pengurangan 57%, tapi waktu penyelesaian bertambah dengan rata-rata 17 detik. Bantuan pengereman berhasil mengurangi jumlah tabrakan, akan tetapi penambahan waktu penyelesaian lintasan tersebut sangat besar.

Hasil pengujian FPS menunjukkan penurunan FPS dengan rata-rata 1,46 saat bantuan pengereman diaktifkan. Perubahan rata-rata FPS sangat kecil, sehingga rata-rata FPS sebelum dan sesudah penggunaan bantuan pengereman tidak menggangu performa pemain.

Hasil bantuan pengereman ini sudah cukup bagus, akan tetapi masih dapat ditingkatkan lagi. Beberapa saran untuk penelitian berikutnyai, antara lain:

1. Gunakan *framework* gim balapan dengan fisika yang lebih bagus. Kekurangan *racing game starter kit* adalah tidak adanya penalti yang mengurangi kecepatan mobil di rumput, sehingga pemain dengan mudah memotong lintasan dengan menembus batas rumput. Akan lebih baik jika mencari framework gim balap berfisika lebih bagus.
2. Gunakan *framework* gim balap bertipe lain. *Racing game starter kit* adalah gim balapan bertipe *sim-racing* yang mencoba mengemulasikan lintasan balap serealistis mungkin. Pengaruh bantuan pengereman kemungkinan akan berbeda pada tipe gim balap lain seperti *kart-racing* dan *arade-racing*.
3. Gunakan algoritme *machine learning* atau struktur jaringan saraf terapan berbeda. Algoritme yang digunakan dalam penelitian ini adalah jaringan saraf terapan dengan struktur dasar. Jaringan saraf terapan dengan struktur lebih kompleks atau algoritme *machine learning* berbeda berpotensi meningkatkan hasil.
4. Tambahkan akselerasi sebagai keluaran bantuan pengereman. Tiga sumber daya roda adalah akselerasi, belokan dan rem. Sehingga jika akselerasi ditambahkan sebagai keluaran JST, kemungkinan keluarannya akan lebih bagus, sehingga pemain pemula hanya perlu membelokkan mobil.
5. Ambil data dari lebih dari satu pemain ahli. Tiap pemain ahli memiliki teknik mengerem yang berbeda. Menambah variasi data latih dan data uji berpotensi meningkatkan hasil JST.

8. DAFTAR PUSTAKA

Cimperman, Rob., 2006. *UAT Defined: A Guide to Practical User Acceptance Testing*. Pearson Education.

Gregory, Jason., 2009. *Game Engine Architecture*. [e-book]. CRC Press. Tersedia di: Google Books <http://booksgoogle.co.id> [Diakses 1 Maret 2019].

Gao, Jerry., Tsao, H.S.J., Wu, Ye., 2003. *Testing and Quality Assurance for Component-based Software*. London: Artech House.

Heyakin, Seymour., 2009. *Neural Network And Learning Machine*. Hamilton: Prentice Hall.

Claypool, K. T., Claypool M., 2007. *On frame rate and player performance in first person shooter games*. [pdf] Multimed Syst. [Diakses 1 Maret 2019].

McConnel, Steve., 2004. *Code Complete, 2nd edition.* Redmond : Microsoft Press.

Read, Paul., Meyer, Mark-Paul., 2000. *Restoration Of Motion Picture Films*. Butterworth-Heinemann Ltd.

Sugawara, Akitomo., 2010. *Gran.Turismo.5.Official.Game.Guide*. [e-book]. CRC Press. Tersedia di: archive.org <http://archive.org> [Diakses 1 Maret 2019].

Powers, David M. W., 2012. *The Problem with Kappa*. Adelaide : Flinders University Of South Australia.

Prasetya, Herlambang Y., 2018. *Penerapan Neural Network Untuk NPC Braking Decision Pada Racing Game*. S1. Universitas Brawijaya.

Unity Technology, 2019. Unity (2019.2). [program komputer] Unity Technology. Tersedia di: <https://unity.com/releases/2019-2> [Diakses 1 Maret 2019]