Laman web jurnal: http://ejournal.stikom-db.ac.id/index.php/processor

Processor: Jurnal Ilmiah Sistem Informasi, Teknologi Informasi dan Sistem Komputer

P-ISSN: 1907-6738 | E-ISSN: 2538-0082

Pengaplikasian Kalman Filter sebagai Pengendali dalam Permainan The Open Racing Car Simulator (TORCS)

Rendy Andrian Yahya[1], Arini[2], Victor Amrizal[3]

[1,2,3] UIN Syarif Hidayatullah Jakarta, Jl. Ir H. Juanda No.95, Cemp. Putih, Kec. Ciputat, Kota Tangerang Selatan, Banten 15412, Indonesia.

ABSTRACT

The Open Racing Car Simulator (TORCS) works as both a playable game and a framework to develop artificial intelligence-based controllers. As a platform for researchers, TORCS has become a platform in controller development with various approaches in artificial intelligence using sensors and actuators provided by the SCR Server via the SimpleDriver architecture. In this research, the author develops a controller using the Kalman Filter, an algorithm to predict and measure states based on previous measurements to determine future trajectory with Rapid Application Development (RAD) chosen as the development method. The test took place in 5 different circuits available from the TORCS installation to compare its performance with another controller. The test result showed that KalmanDriver, the developed controller, has an adequate average performance close enough to results in a non-noisy setting and against another controller in noisy setting, both in terms of fastest times and top speeds.
Keywords: Artificial Intelligence, Kalman Filter, Controller, TORCS, Simulation.

ABSTRAK

The Open Racing Car Simulator (TORCS) merupakan sebuah permainan yang juga menjadi landasan untuk mengembangkan dan menguji model pengendali berbasis kecerdasan buatan. Sebagai platform untuk komunitas ilmiah, TORCS telah menjadi bahan riset dalam pengembangan pengendali dengan berbagai pendekatan kecerdasan buatan menggunakan permodelan sensor dan aktuator yang disediakan arsitektur SimpleDriver dalam SCR Server. Dalam penelitian ini, penulis bertujuan untuk mengembangkan pengendali yang mengimplementasikan Kalman Filter sebagai pendekatan yang digunakan dan menggunakan metode pengembangan RAD (Rapid Application Development), dengan pengujian yang menggunakan 5 sirkuit yang tersedia dalam permainan serta dibandingkan performanya dengan pengendali lainnya. Hasil dari pengujian menunjukkan bahwa KalmanDriver, pengendali yang dikembangkan untuk penelitian ini, memiliki rerata performa yang mendekati pengujian tanpa pengaturan derau serta pengendali lainnya dalam pengaturan derau, baik dalam hal catatan waktu ataupun raihan kecepatan tinggi.
Kata Kunci: Kecerdasan Buatan, Kalman Filter, Pengendali, TORCS, Simulasi.

1. PENDAHULUAN

Sejak awal perkembangkannya di tahun 1956, konsep kecerdasan buatan telah mengalami beberapa perkembangan agar bisa diterapkan pada keseharian pengguna [28]. Konsep kecerdasan buatan sendiri merupakan bidang ilmu yang fokus pada proses berpikir dan tingkah laku [27], dan telah diaplikasikan pada berbagai bidang, salah satunya pada permainan video. Penggunaan kecerdasan buatan dalam permainan juga mengalami perkembangan sejalan dengan perkembangan kecerdasan buatan pada umumnya, dengan pengimplementasian algoritma berbeda yang dikembangkan oleh para pengembang permainan, mulai dari *state machine* sampai beberapa algoritma kecerdasan buatan seperti *pathfinding*, *neural network*, dan lainnya.

Pac-Man milik Midway Games West Inc. yang dirilis pada tahun 1979 dan *Pong* yang dirilis oleh Atari pada tahun 1972 merupakan dua permainan video pertama yang menggunakan kecerdasan buatan. Dalam *Pac-Man*, empat hantu yang dikendalikan oleh kecerdasan buatan menggunakan teknik simpel yang berupa *state machine*, dimana dalam tiap *state* terdiri dari mengambil rute acak tiap persimpangan dalam arena permainan sebagai pergerakan standar, mengejar pemain, dan/atau melarikan diri [6]. Sementara dalam *Pong*, tongkat lawan berusaha untuk menepis bola dari gawangnya ke tongkat pemain. Pergerakan lawan ini menggunakan perhitungan simpel yang bisa mengkalkulasikan ketinggian dimana bola akan masuk ke gawang dan menggerakan tongkat tersebut secepatnya, tergantung tingkat kesulitan yang dipilih oleh pemain [9].

Gambar 1. Tangkapan Layar dari Permainan *Pole Position* Dimana Pemain Berhadapan dengan Pemain Lawan dalam Perlombaan [20].

Kecerdasan buatan dalam permainan video berkembang pada pertengahan 1990an, dimana teknik yang digunakan mulai bervariasi. *Goldeneye 007* (Rare Ltd., 1997) dan *Metal Gear Solid* (Konami Corporation, 1998) menggunakan sistem *sense simulation* dimana karakter bisa melihat teman-teman mereka dan akan tahu jika mereka terbunuh. Permainan *Real Time Strategy Warcraft* (Blizzard Entertainment, 1994) menggunakan teknik *pathfinding* untuk pemain komputernya, dan *Creatures* (Cyberlife Technology Ltd., 1997) menggunakan sistem kecerdasan yang paling rumit, yaitu otak berbasis *neural network* untuk tiap makhluk dalam permainan tersebut [5]. Selain itu, kecerdasan buatan dalam permainan menembak *Counter Strike* (Valve Software, 2000) terprogram untuk memiliki kemampuan untuk mengemulasikan semirip mungkin pergerakan yang biasa dilakukan oleh para pemain menurut Khoo dan Rubek dalam [16].

Kecerdasan Buatan dalam *genre* balap pertama kali diaplikasikan pada *Pole Position*, permainan milik Namco yang dirilis pada tahun 1982. Dalam permainan ini, jika pemain berhasil menyelesaikan kualifikasi dalam waktu yang ditentukan, pemain akan dihadapkan dengan tujuh mobil lainnya yang semuanya dikendalikan oleh komputer [19]. Beberapa game ber-genre racing lainnya yang memiliki algoritma kecerdasan buatan yang lebih canggih termasuk diantaranya *NASCAR Thunder 2002* milik EA Sports memiliki lawan komputer dengan kepribadian dan gaya mengendara yang berbeda. Tergantung dari bagaimana pemain mencoba untuk menyalip salah satu mobil lawan ini, ada kemungkinan mobil yang tersalip ini untuk melakukan manuver agresif terhadap pemain [9]. Selain itu, *Forza Motorsport* dari Microsoft Game Studio memiliki fitur "*drivatars*" yang mana pemain memainkan sekumpulan bagian sirkuit yang didesain untuk menampung sekumpulan tantangan yang representatif. Dalam proses ini, permainan merekam jalur yang dilalui dalam tiap segmen dan akan diimplementasikan pada "*drivatar*" untuk sirkuit baru dengan cara memisahkan jalur yang dilalui pada tiap segmen sebuah sirkuit [21].

Dalam pengembangan kecerdasan buatan untuk permainan beraliran balap, *The Open Racing Car Simulator* (TORCS) [3], selain menjadi permainan yang bisa dimainkan oleh pemain manusia, juga merupakan sebuah landasan untuk mengembangkan dan menguji model kecerdasan buatan untuk pengemudi swatantra yang juga dapat disebut sebagai pengendali [16]. Sebagai landasan untuk komunitas ilmiah, TORCS telah menjadi bahan riset dalam pengembangan pengendali dengan algoritma kecerdasan buatan yang bervariasi yang ditulis dalam beberapa bahasa pemrograman, serta telah digunakan secara kompetitif dalam beberapa ajang bertemakan kecerdasan buatan, diantaranya ACM Genetic and Evolutionary Computation Conference (GECCO), dan IEEE Symposium on Computational Intelligence and Games (CIG) [4], serta masih aktif digunakan untuk kompetisi berbasis komunitas TORCS Endurance Championship dimana peserta mengirimkan pengendali mereka pada satu server untuk diperlombakan pada komputer masing-masing peserta [1].

Gambar 2. Tangkapan layar dari permainan *The Open Racing Car Simulator* (TORCS) [26]

Sebagai landasan untuk pengembangan pemain berbasis kecerdasan buatan, telah dikembangkan pemain kecerdasan buatan dalam TORCS yang menggunakan algoritma kecerdasan buatan yang bervariasi, mulai dari *neural network*, pemrograman genetika, serta *computer vision*. Arsitektur permodelan sensor dan aktuator digunakan sebagai landasan pengembangan pengendali baru dalam TORCS, dimana permodelan sensor dan aktuator disediakan pada arsitektur SCR Server yang menjadi ekstensi TORCS. Pengendali menerima lingkungan perlombaan melalui sensor yang memberikan informasi tentang ruang lingkup permainan seperti diantaranya sensor lintasan (yang terdiri dari 19 sensor untuk mendeteksi jarak terhadap tepi lintasan), sensor pemain lawan (36 sensor pendeteksi pemain lawan), dan sensor kecepatan (mengembalikan kecepatan mobil dalam arah sumbu tertentu), serta kondisi sementara dari perlombaan seperti waktu putaran berjalan serta posisi dalam perlombaan. Pengendali juga mengendalikan mobil dalam permainan melalui model aktuator yang tersedia, seperti setir kemudi, pedal gas, pedal rem, serta girboks untuk pergantian gigi [5].

Selama delapan tahun terakhir (sejak 2012), permodelan sensor dan aktuator dalam SCR Server menjadi landasan mayoritas para peneliti dalam mengembangkan pengendali dengan beberapa pendekatan kecerdasan buatan yang berbeda pada beberapa jurnal, seperti diantaranya [23] dimana pengendali mengandalkan penglihatan visual untuk mengendalikan mobil untuk mengitari sirkuit dengan sukses setelah beberapa iterasi serta memberikan visualisasi pergerakan yang dilalui oleh modul pengajar (pemain) dan modul siswa (pengendali) dalam satu sirkuit penuh, [8] dimana vektor fitur terkomputasikan oleh pengendali *recurrent neural network* (RNN) dengan metode *Neuroevolution* (NE), dan [17] dimana pendekatan persepsi langsung dilakukan untuk mengestimasikan kelayakan untuk mengemudi menggunakan *convolutional neural network* menggunakan rekaman berkendara virtual manusia dalam selama 12 jam. Selain itu, [13] serta [16] mengembangkan pengendali berbasis logika *fuzzy* dengan pendekatan yang berbeda; arsitektur dalam [13] bertujuan untuk mengimplementasikan strategi penghindaran dan penyalipan pemain lawan dengan strategi penghadangan berbeda sementara arsitektur dalam [16] mengkalkulasikan input kemudi dan komputasi kecepatan target secara bersamaan dalam

mengemudikan mobil dimana kedua fungsi tersebut dikembangkan sebagai dua sub-pengendali terpisah [5] mempraktekan teknik penyalipan dan penghadangan pengendali yang dikembangkan menggunakan sirkuit dan pemain lawan yang disediakan TORCS. Selain itu, terdapat beberapa pendekatan yang diperlombakan dalam Simulated Car Racing Championship pada konferensi GECCO 2013, kompetisi berbasis TORCS sebagai landasan untuk pengendali yang dikembangkan. Beberapa diantaranya termasuk *Hill climbing* dan *Neural network* [10], *evolutionary algorithm* [24], algoritma heuristik [15], dan *Gene regulatory network* [25].

Kalman Filter merupakan algoritma untuk memprediksi pergerakan selanjutnya dari sebuah sistem berdasarkan perhitungan sebelumnya yang dikembangkan oleh Rudolf Emil Kalman pada tahun 1960an. Fungsi dari algoritma ini adalah untuk memprediksi estimasi lokasi obyek selanjutnya, mengurangi derau yang disebabkan oleh deteksi yang tidak akurat, serta memfasilitasi proses pengelompokan obyek pada jalurnya [18]. Kalman Filter berawal sebagai metode untuk menyelesaikan permasalahan navigasi dalam Apollo Project dan telah diaplikasikan sebagai estimator dalam banyak bidang, salah satu diantaranya adalah navigasi [12]; pada [7], Kalman Filter digunakan sebagai penentu pergerakan dari sebuah kendaraan, dalam hal ini sebuah traktor. Penelitian tersebut menggunakan sensor GPS yang disematkan pada traktor agar kendaraan tersebut dapat menentukan pergerakan dengan permodelan kinematik yang telah ditentukan.

Dengan mempertimbangkan penggunaan Kalman Filter sebagai alat navigasi serta fungsionalitas dari TORCS sebagai landasan pengembangan pengendali berbasis kecerdasan buatan, penulis melakukan penelitian ini dengan tujuan: 1) untuk memperlihatkan cara kerja pengembangan pengendali dalam TORCS; 2) untuk mengimplementasikan Kalman Filter sebagai pengendali untuk TORCS; 3) untuk melihat performa pengendali tersebut dibandingkan pengendali lainnya dalam pengujian pada lima sirkuit yang disediakan TORCS.

2. TINJAUAN PUSTAKA

2.1. Kalman Filter

Kalman Filter merupakan sebuah algoritma yang digunakan sebagai metode untuk memprediksi perhitungan pada waktu selanjutnya dari sebuah sistem berdasarkan perhitungan sebelumnya. Kalman Filter dinamai setelah pengembang utama dari algoritma ini, yaitu Rudolf Emil Kalman, dan dikenal karena kebutuhan komputasionalnya yang relatif kecil, memiliki sifat rekursif yang elegan, dan statusnya sebagai estimator optimal untuk sistem linier satu dimensi dengan statistik kesalahan Gaussian. Beberapa kegunaan dari Kaiman Filter diantaranya untuk melancarkan derau dan menyediakan estimasi dari parameter yang digunakan [22]. Nama lain dari Kalman Filter adalah "*Linear Quadratic Estimation*" (LQE) [14].

Kalman Filter pertama kali digunakan pada tahun 1960an sebagai metode untuk menentukan estimasi dan prediksi dalam navigasi Apollo Project, dan telah teraplikasikan dalam beberapa bidang, diantaranya navigasi, keuangan [12], pengolahan citra, dan *computer vision* [22].

Kalman Filter dirancang untuk beroperasi pada sistem dalam lingkup linear. Algoritma ini menggunakan sebuah prediksi yang diikuti dengan koreksi untuk menentukan estimasi perhitungan dari filter yang sedang digunakan, yang juga bisa disebut sebagai *predictor-corrector* atau *prediction-update* (prediksi dan perbaruan). Dengan menggunakan informasi mengenai dinamika perhitungan, filter akan "bergerak" kedepan dan memprediksi bagaimana perhitungan selanjutnya berjalan. Bagian perbaruan lalu melibatkan perbandingan sebuah pengukuran dengan prediksi perbandingan berdasarkan perhitungan yang terprediksi.

Rumus untuk model prediksi terdiri dari:

$$\hat{x}_{k|k-1} = F_{k-1}X_{k-1} + B_{k-1}\mu + v \quad (1a)$$

$$P_{k|k-1} = F_{k-1}P_{k-1}F^{T}{}_{k-1} + Q_{k-1} \quad (1b)$$

Sementara rumus untuk model pengukuran dan perbaruan terdiri dari:

$$z_k = H\hat{x}_{k|k-1} + w \quad (2a)$$

$$Y_k = z_k - \hat{x}_{k|k-1} \quad (2b)$$

$$K_k = P_{k|k-1}H^T\left(HP_{k|k-1}H^T + R_k\right)^{-1} \quad (2c)$$

$$\hat{x}_k = \hat{x}_k + K_kY_k \quad (2d)$$

$$P_k = (I - K_kH)P_{k|k-1} \quad (2e)$$

2.2. The Open Racing Car Simulator (TORCS)

The Open Racing Car Simulator, disingkat sebagai TORCS, adalah simulator balap mobil *multi-platform* portabel. Simulator ini digunakan sebagai baik simulator balap mobil untuk pemain manusia, sebagai permainan balap untuk kecerdasan buatan, dan sebagai

platform riset. Simulator ini berjalan pada sistem operasi Linux (seluruh arsitektur, 32 dan 64 bit, serta little dan big endian), FreeBSD, OpenSolaris, MacOSX, dan Windows (32 dan 64bit). TORCS dikembangkan oleh Eric Espié and Christophe Guionneau, serta diteruskan pengembangannya oleh Bernhard Wymann. Beberapa bagian tambahan juga ditambahkan oleh kontributor lainnya pada halaman "*Credits*" dari situs utama TORCS, sementara *source code* dari TORCS dilisensikan dibawah GPL ("*Open Source*").

TORCS memiliki pilihan mobil, sirkuit, dan lawan komputer yang berbeda, dan memperbolehkan para pemainnya untuk menggunakan *joystick* atau *steering wheel*, selain tetikus dan/atau *keyboard*, yang terdukung oleh sistem operasi yang digunakan. Fitur grafis yang dimiliki TORCS antara lain pencahayaan, asap, *skidmark*, dan diska rem yang menyala, sementara fitur simulasinya termasuk permodelan kerusakan simpel, tabrakan, ban (termasuk sistem suspensi), aerodinamika (*ground effect*, sayap belakang), dan lebih banyak lagi. Selain itu, permainan ini juga memiliki banyak tipe balapan yang bervariasi, mulai dari mode latihan (*practice*) sampai kejuaraan (*championship*), serta dukungan untuk permainan *split-screen* yang dapat memuat empat pemain manusia [1].

TORCS juga digunakan sebagai media untuk beberapa kompetisi berbasis kecerdasan buatan, beberapa diantaranya termasuk TORCS Endurance Championship, Simulated Car Racing Championship (Wymann dan Espié, 2004), serta kompetisi yang diadakan oleh IEEE Congress on Evolutionary Computation (CEC), ACM Genetic and Evolutionary Computation Conference (GECCO), dan IEEE Symposium on Computational Intelligence and Games (CIG) [4].

2.3. Pengendali

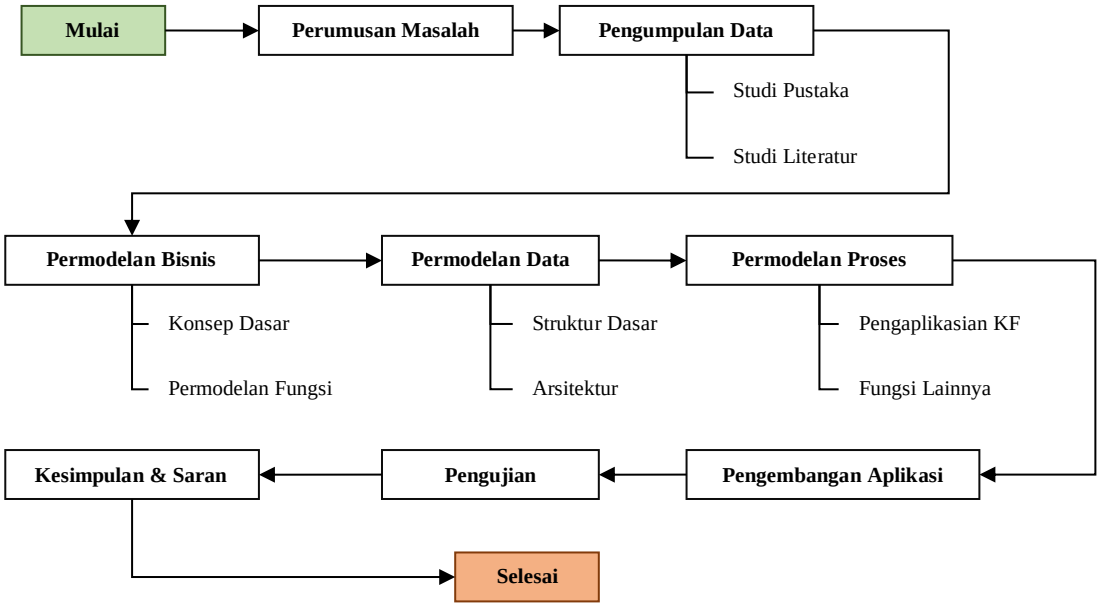
Berdasarkan definisi dari [16], sebuah pengendali terdefinisikan sebagai permodelan Kecerdasan Buatan untuk pengemudi swatantra dalam *The Open Racing Car Simulator* (TORCS). Sebuah pengendali juga dapat disebut sebagai bot. Pengendali diimplementasikan sebagai bagian perangkat lunak yang terpisah dengan arsitektur modular, menjadikannya mudah untuk mengembangkan pengendali yang baru dan diimplementasikan dalam permainan.
Pada dasarnya, pengendali memiliki beberapa modul yang mengendalikan mobil saat perlombaan berlangsung: dalam modul pengendalian kecepatan, pengendali mengkalkulasikan kecepatan dan posisi gigi yang diinginkan. Pengendali akan menaikan gigi jika nilai rpm melebihi jumlah yang ditentukan, dan sebaliknya akan menurunkan gigi jika nilai rpm kurang dari yang ditentukan. Modul ini diaktifkan untuk mengatur kecepatan mobil saat memasuki tikungan. Selain itu, modul ABS (*Antilock Braking System*) dan TCL (*Traction Control*) melindungi mobil dari slipping, dan pengendalian arah mobil dilakukan oleh modul steering. Terakhir, jika mobil berada dalam posisi terjebak, pengendali akan menggunakan strategi pengendalian yang berbeda. Dengan kompleksitas tersebut, banyak pendekatan kecerdasan komputer yang digunakan dalam pengembangan pengendali, contohnya pengendali berbasis *fuzzy* untuk menentukan nilai kecepatan yang diinginkan dengan aturan yang dioptimisasikan menggunakan algoritma genetika [11].

Tiap waktu simulasi, pengendali harus membuat sebuah keputusan, misalkan untuk mengendalikan mobil dalam lintasan dan mengatur target kecepatan. Permasalahan ini akan menjadi lebih sulit ketika mobil mendekati sebuah tikungan dimana pengendali harus memilih antara melaju lebih cepat dan mencoba untuk tidak terpelintir keluar lintasan. Lain kata, pengendali mencoba menggerakkan mobil menuju tepi lintasan terjauh untuk menghindari tabrakan [16].

Jurnal-jurnal dengan topik pengembangan pengendali dalam TORCS yang digunakan sebagai landasan telah diterbitkan antara tahun 2012-2016 dengan pendekatan kecerdasan buatan yang berbeda.

3. METODOLOGI PENELITIAN

Untuk penelitian ini, metodologi yang digunakan adalah *Rapid Application Development* (RAD) yang menjamin waktu pengerjaan singkat dan simpel tanpa membutuhkan sumber daya yang besar. Tahap pengerjaan meliputi permodelan konsep pengendali, permodelan arsitektur SimpleDriver, pengaplikasian Kalman Filter dalam pengendali, pengembangan, dan pengujian, berdasarkan pendekatan yang digunakan pada [7], dimana permodelan kinematik kendaraan terdiri dari modulus kecepatan, kemudi, sudut, serta posisi kendaraan dalam koordinat x dan y.



Gambar 3. Visualisasi Tahapan Metodologi yang Digunakan dalam Penelitian

4. HASIL DAN PEMBAHASAN

4.1. Permodelan Bisnis

KalmanDriver merupakan pengendali untuk *The Open Racing Car Simulator* (TORCS) yang akan dikembangkan untuk penelitian ini. Sesuai dengan namanya, KalmanDriver mengaplikasikan Kalman Filter untuk mengendalikan pergerakan mobil dalam permainan. Proyek ini berbasis Java dan pengerjaannya berdasarkan SimpleDriver dalam paket SCR Server, dimana pengendali tersebut merupakan implementasi mendasar untuk pengembangan pengendali berbasis SCR Server yang juga menggunakan permodelan sensor serta aktuator dalam [5].

Fitur utama dari KalmanDriver adalah input kemudi yang pergerakannya terestimasi melalui Kalman Filter, sesuai dengan namanya. Jarak dan titik target akan dihitung berdasarkan keluaran sensor yang ada pada sudut dari sensor yang digunakan, dan dijadikan bahan pengukuran untuk sistem filter yang dikembangkan. Hasil estimasi berdasarkan pengukuran akan digunakan pada model prediksi dalam filter yang akan menentukan estimasi selanjutnya. Penanganan derau dalam filter juga ikut terimplementasikan sebagai matriks pengukuran derau untuk menghasilkan estimasi lebih dekat dari pengukuran tanpa derau.

Selain fungsi diatas, beberapa fungsi lainnya yang digunakan termasuk diantaranya pergantian gigi, serta pengendalian kecepatan. Seluruh pengendalian mobil yang dilakukan pengendali diselesaikan oleh pembacaan sensor dan penggunaan aktuator sesuai dengan penjelasan sebelumnya, dengan komputasi yang ditentukan pada pengembangan.

4.2. Permodelan Data

Aplikasi SCR Server (SCR kepanjangan dari *Simulated Car Racing*) berfungsi sebagai perangkat lunak tambahan untuk arsitektur asli TORCS dalam tiga proses: pertama, aplikasi ini menstrukturkan TORCS sebagai aplikasi *client-server* dimana pengendali atau bot dijalankan sebagai proses eksternal yang tersambung ke server perlombaan melalui koneksi UDP. Kedua, aplikasi ini melakukan perhitungan secara aktual dimana pada tiap *game tick* (kurang lebih sama dengan 20 milidetik dalam permainan) *server* mengirimkan input sensor berjalan ke tiap pengendali dan nantinya server tersebut akan menunggu selama 10 milidetik waktu nyata untuk menerima tindakan dari controller tersebut. Akan tetapi, jika tidak ada tindakan, simulasi berlanjut dan tindakan sebelumnya digunakan. Terakhir, aplikasi ini menciptakan pemisahan fisik antara *source code* dari pengemudi dan server perlombaan yang membangun lapisan abstraksi, sebuah model sensor and aktuator, yang memberikan kebebasan dalam penggunaan bahasa pemrograman untuk sebuah pengendali dan membatasi akses hanya untuk informasi yang terdefinisikan oleh pengembang pengendali tersebut. [5]

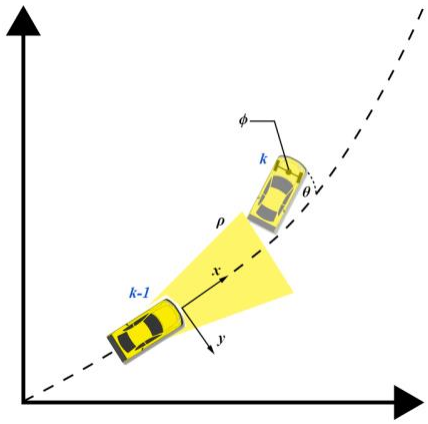
Secara keseluruhan, paket SCR Server menyediakan beberapa kelas untuk menopang pengembangan pengendali untuk TORCS. Kelas yang ada mengimplementasikan arsitektur *client-server* dalam permainan serta permodelan sensor dan aktuator yang digunakan dalam permainan. Pengerjaan pengendali KalmanDriver berdasarkan implementasi dari kelas SimpleDriver yang merupakan implementasi dasar pengendali menggunakan permodelan sensor dan aktuator sesuai [5]. Dalam kelas ini, didefinisikan beberapa fungsi yang digunakan sebagai implementasi dasar dari pengendali. Tabel 1 dibawah menjelaskan secara rinci fungsi-fungsi dasar yang tersedia pada SimpleDriver.

Tabel 1. Fungsi-Fungsi Dasar yang ada pada Kelas Simpledriver

Fungsi	**Deskripsi**
public void reset()	Fungsi yang dijalankan saat perlombaan diulang atau restart.
public void shutdown()	Fungsi yang dijalankan saat perlombaan berakhir.
private int getGear(SensorModel sensors)	Implementasi pergantian gigi berdasarkan pembacaan nilai RPM mobil.
private float getSteer(SensorModel sensors)	Implementasi kemudi dalam mengendalikan mobil.
private float getAccel(SensorModel sensors)	Implementasi pengaturan kecepatan dalam mengendalikan mobil.
public Action control(SensorModel sensors)	Implementasi model aktuator pengendali untuk input dalam permainan berdasarkan fungsi pengendalian mobil.
private float filterABS(SensorModel sensors,float brake)	Implementasi ABS (*Antilock Braking System*) pada proses pengereman.
float clutching(SensorModel sensors, float clutch)	Implementasi kopling pada proses input kopling dalam permainan.
public float[] initAngles()	Inisialisasi sudut dari 19 sensor lintasan.

4.3. Permodelan Proses

4.3.1. Permodelan Kalman Filter



Gambar 4. Visualisasi Permodelan Kinematik untuk Pengaplikasian Kalman Filter pada Pengendali.

Sistem untuk filter dimodelkan untuk mengestimasikan pergerakan mobil dalam permainan berdasarkan pengukuran yang dimasukan serta permodelan derau sistem yang digunakan. Hasil estimasi dari sistem filter nantinya akan digunakan sebagai input kemudi pengendali dalam permainan. Untuk memodelkan sistem untuk filter yang akan dikembangkan, penulis memodelkan permodelan kinematik dasar dari pergerakan mobil berdasarkan permodelan sensor yang tersedia. Mengikuti implementasi [7], model kinematik untuk sistem mengikuti model sepeda roda tiga dimana dua ban depan mengendalikan pergerakan dari mobil (kemudi) dan dua ban belakang hanya bergerak lurus. Input sistem dalam model ini terdiri dari modulus kecepatan ρ dan input kemudi α, diekspresikan sebagai matriks vektor waktu $\hat{x}$ sebagai:

$$\hat{x}_{k|k-1} = [x \quad y \quad \theta \quad \rho \quad \phi]^T \tag{3}$$

Dengan asumsi bahwa sistem tidak berada dalam kondisi licin, persamaan masing-masing model kinematik terdiri dari:

$$x = \rho \cos\theta \tag{4a}$$

$$y = \rho \sin\theta \tag{4b}$$

$$\phi = -\tan^{-1}\left(\frac{y}{x}\right) \tag{4c}$$

Dengan menggunakan seluruh sensor lintasan dengan pengecualian sensor dengan sudut -90 dan 90 derajat serta -75 dan 75 derajat (15 dari 19 sensor lintasan yang digunakan), x dan y merupakan hasil penambahan vektor dari jarak dan titik tiap sensor dengan rumus:

$$x = \rho_1 \cos\theta_1 + \rho_2 \cos\theta_2 + \cdots + \rho_n \cos\theta_n \tag{5a}$$

$$y = \rho_1 \sin\theta_1 + \rho_2 \sin\theta_2 + \cdots + \rho_n \sin\theta_n \tag{5b}$$

ρ_n dan θ_n masing-masing didefinisikan dengan rumus:

$$\rho_n = |ts_n{}^3| \tag{6a}$$

$$\theta_n = 2rad_n - \frac{\delta \sin|\delta|}{2} \tag{6b}$$

Dikarenakan penggunaan sensor lintasan untuk perhitungan seperti yang dijelaskan, n yang digunakan dimulai dari sensor kedua sampai sensor ke-16, artinya $n = 1$ menandakan sensor ketiga, $n = 2$ menandakan sensor keempat, dan seterusnya.

Terakhir, untuk memenuhi model prediksi dari filter, matriks F didefinisikan dengan:

$$F_{k-1} = \begin{bmatrix} 1 & 0 & 0 & \Delta t \cdot \cos\theta_{k-1} \\ 0 & 1 & 0 & \Delta t \cdot \sin\theta_{k-1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Dengan ini, model kinematik dasar tiap vektor keadaan yang dihasilkan masing-masing berupa:

$$x_{k|k-1} = x_{k-1} + \rho_{k-1} \cdot \Delta t \cdot cos\, \theta_{k-1} \quad (8a)$$

$$y_{k|k-1} = y_{k-1} + \rho_{k-1} \cdot \Delta t \cdot sin\, \theta_{k-1} \quad (8b)$$

$$\theta_{k|k-1} = \theta_{k-1} \quad (8c)$$

$$\rho_{k|k-1} = \rho_{k-1} \quad (8d)$$

Selebihnya untuk sistem prediksi, penulis mendefinisikan B dan μ sebagai 0 dikarenakan sistem tidak memiliki pengaruh luar. Ukuran matriks P yang terdefinisikan untuk sistem adalah 4x4, dan ν diasumsikan 0 pada sistem ini. Isi matriks Q didefinisikan dengan:

$$Q_{k-1} = \begin{bmatrix} q_{11_k} & q_{12_k} & q_{13_k} & q_{14_k} \\ q_{21_k} & q_{22_k} & q_{23_k} & q_{24_k} \\ q_{31_k} & q_{32_k} & q_{33_k} & q_{34_k} \\ q_{41_k} & q_{42_k} & q_{43_k} & q_{44_k} \end{bmatrix} \quad (9)$$

Untuk proses pengukuran, input untuk matriks z_k terdiri dari koordinat x dan y, sudut θ, serta jarak ρ, dengan ini input vektor waktu $\hat{x}$ dan z sama. Selanjutnya matriks H yang digunakan pada sistem ini sama persis dengan matriks identitas, bisa diekspresikan sebagai:

$$H = I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

Nilai w yang digunakan merupakan derau yang berasal dari sensor lintasan yang digunakan. Derau dari tiap sensor lintasan memiliki standar deviasi sampai 10% jangkauan maksimal (kemungkinan tertinggi pembacaan bisa sampai 220m). Perlu diingat bahwa pengukuran sudut titk target θ tidak memiliki model derau pengukuran karena hanya menghitung titik target berdasarkan posisi sensor lintasan dan orientasi arah mobil terhadap sumbu positif lintasan, jadi θ selalu tepat perhitungannya.

Pada perhitungan Kalman Gain sesuai persamaan 2c, bagian $\left(P_{k|k-1} + R\right)$ juga dapat dikenal sebagai matriks inovasi. Matriks R_k yang digunakan memiliki ukuran 4 x 4, sehingga dapat ditulis sebagai:

$$R_k = \begin{bmatrix} r_{11_k} & r_{12_k} & r_{13_k} & r_{14_k} \\ r_{21_k} & r_{22_k} & r_{23_k} & r_{24_k} \\ r_{31_k} & r_{32_k} & r_{33_k} & r_{34_k} \\ r_{41_k} & r_{42_k} & r_{43_k} & r_{44_k} \end{bmatrix} \quad (11)$$

Hasil dari $\hat{x}_k$ dijadikan input untuk kemudi dalam pengendali dan digunakan sebagai bahan perhitungan untuk waktu selanjutnya.

4.3.2. Fungsi Pergantian Gigi

Fungsi untuk pergantian gigi dalam permainan diselesaikan dengan mengecek nilai RPM (rotasi per menit) yang sedang dicapai. Jika nilai RPM mencukupi untuk menaikan gigi, maka pengendali akan pindah ke gigi selanjutnya. Sebaliknya, pengendali akan menurunkan gigi jika nilai RPM kurang. Pengendali yang dikembangkan memiliki enam gigi dengan nilai konstan RPM yang ditentukan untuk proses perpindahan gigi tersebut.

Tabel 2. Nilai RPM maksimal dan minimal tiap gigi untuk pergantian gigi dalam pengendali.

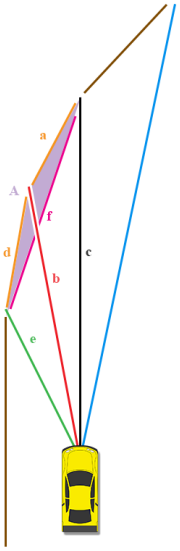
Gigi ke-*n*	**Nilai rpm Minimal**	**Nilai rpm Maksimal**
Gigi pertama	0	8000
Gigi kedua	3000	8000
Gigi ketiga	3500	8200
Gigi keempat	4000	8200
Gigi kelima	4500	8200
Gigi keenam	5000	0

Pada pengendali Kalman Driver, penulis memilih nilai maksimal dan minimal RPM tiap gigi berdasarkan pengalaman penulis dalam permainan. Peneliti berasumsi bahwa nilai RPM yang terpilih merefleksikan titik pergantian gigi yang optimal dalam mengemudikan mobil.

4.3.3. Fungsi Pengaturan Kecepatan

Sesuai standar SimpleDriver dalam mengatur kecepatan, pengendali akan mencapai kecepatan target berdasarkan pembacaan tiga sensor lintasan berbeda dari 19 yang tersedia: sensor tengah (sensor kesepuluh), sensor sebelah kiri (sensor kesembilan), dan sensor

sebelah kanan (sensor kesebelas), masing-masing berjarak lima derajat antara satu sama lain. Tergantung dari pembacaan sensor, pengendali dapat melaju dengan kecepatan penuh atau mengendalikan kecepatan sesuai radius tikungan yang dikalkulasikan. Akan tetapi, jika mobil keluar lintasan, pengendali akan memasukan input akselerasi sedang. Hal ini untuk menghindari kondisi tidak stabil saat mencoba kembali ke lintasan dengan asumsi medan untuk luar lintasan bukan medan aspal (seperti rerumputan dan jebakan pasir).



Gambar 5. Visualisasi Pengukuran Radius Tikungan.

Pada pengendali yang dikembangkan, peneliti menggunakan pendekatan baru untuk perhitungan radius tikungan dalam mengkalkulasikan kecepatan target yang akan dicapai. Pendekatan yang digunakan adalah untuk mengkalkulasikan radius berdasarkan sisi luar tikungan seperti gambar diatas: sudut biru menandakan arah tikungan yang akan dituju, dan sudut warna merah dan hijau menandakan bahan perhitungan yang akan digunakan. Pendekatan baru ini digunakan untuk memperoleh kecepatan target yang lebih tinggi dan kompetitif.

Berdasarkan permodelan tersebut, radius sebuah tikungan ditentukan menggunakan metode aturan kosinus untuk mencari sisi antara dua sensor tepi lintasan yang digunakan. Tiga sensor lintasan membentuk tiga buah segitiga dengan masing-masing sisi terdiri dari sisi tertinggi (sensor tengah, garis hitam pada Gambar 4), sisi sedang (satu sensor setelah sensor tengah, direpresentasikan sebagai garis merah dalam gambar), serta sisi terendah (dua sensor setelah sensor tengah, direpresentasikan sebagai garis hijau dalam gambar). Ketiga segitiga tersebut terdiri dari sisi tertinggi dan sedang ($\triangle CAB$) dengan sudut segitiga aa, sisi sedang dan terendah ($\triangle BDE$) dengan sudut segitiga dd (dihitung dari selisih antara aa dan ff), serta sisi tertinggi dan terendah ($\triangle CFE$) dengan sudut ff. Dengan ini, sisi a, d, dan f dikomputasikan berdasarkan rumus:

$$a = \sqrt{b^2 + c^2 - 2bc\,cos(aa)} \tag{12a}$$

$$d = \sqrt{b^2 + e^2 - 2be\,cos(dd)} \tag{12b}$$

$$f = \sqrt{c^2 + e^2 - 2ce\,cos(ff)} \tag{12c}$$

Hasil perhitungan sisi a, d, dan f juga akan membentuk segitiga baru $\triangle ADF$ yang merupakan segitiga terluar yang mengkalkulasikan radius akhir sebuah tikungan. Perhitungan luas $\triangle ADF$ hanya dilakukan jika a dan d dijumlahkan lebih besar daripada f, dan secara pemrograman digunakan untuk menghindari penggunaan berlebihan dari fungsi trigonometri. Perhitungan ini dilakukan menggunakan rumus:

$$s = \frac{a+d+f}{2} \tag{13}$$

Hasil s kemudian digunakan pada rumus berikutnya berdasarkan Teorema Heron yaitu:

$$A = \sqrt{s(s-a)(s-d)(s-f)} \tag{14}$$

Terakhir, luas A digunakan untuk menghitung *circumradius* dari segitiga tersebut sehingga hasilnya akan menjadi radius akhir tikungan yang dilalui. Dari luas A dan sisi a, d, dan f, perhitungan radius diselesaikan dengan rumus:

$$r = min\left(\frac{adf}{4A}, 10000\right) \tag{15}$$

Selanjutnya, setelah perhitungan radius tereselesaikan, kecepatan target dalam tikungan dapat dihitung melalui rumus:

$$v' = 5\sqrt{\mu G(r + ts_{10})} \quad (16)$$

Nilai 5 pada persamaan diatas diatur sedemikian rupa oleh penulis untuk mendapatkan hasil kecepatan target yang tidak hanya cepat tapi juga aman tanpa berakhir keluar lintasan karena terlalu cepat.

Input untuk akselerasi dan rem diselesaikan pada persamaan:

$$a = \frac{2}{\left(1+e^{(v-v')}\right)-1} \quad (17)$$

Melalui persamaan eksponensial seperti diatas, pengendali akan mengatur kecepatannya dalam fungsi ini; jika v lebih kecil dari v' maka pengendali akan menghasilkan input akselerasi. Sebaliknya (dengan nilai akselerasi negatif), pengendali akan melakukan pengereman.

4.3.4. Fungsi Kemudi

Fungsi kemudi memiliki dua metode berbeda untuk menentukan input kemudi yang digunakan pengendali, yaitu kemudi sensor berdasarkan sensor lintasan dan kemudi lintasan berdasarkan posisi mobil dalam lintasan. Kemudi sensor digunakan saat mobil berada di dalam lintasan, dan kemudi lintasan akan digunakan ketika mobil berada di luar.
Kemudi sensor menggunakan keluaran dari pembacaan sensor dan menentukan jarak serta titik target searah dengan arah mobil dan bergerak ke arah dengan ruang yang kosong. Komputasi posisi target untuk kemudi sensor kemudian dijadikan bahan pengukuran untuk model filter yang telah dijelaskan sebelumnya, dimana hasil pengukuran menjadi input akhir kemudi sesuai dengan persamaan:

$$\phi = \frac{-\tan^{-1}\left(\frac{y}{x}\right)}{0.366519} \quad (18)$$

Nilai 0.366519 yang digunakan pada persamaan diatas merupakan nilai konstan sudut maksimal kemudi dalam pengendali yang diekspresikan dalam satuan radian, atau yang dapat dikenal sebagai konstan *steerlock* dalam pengendali.

Jika mobil keluar lintasan, kemudi lintasan akan digunakan untuk mengarahkan mobil kembali kedalam lintasan menggantikan kemudi sensor yang digunakan. Kemudi ini digunakan karena keluaran sensor lintasan tidak tersedia pada kondisi tersebut. Perumusan yang digunakan adalah:

$$\phi = -\frac{\frac{\tan^{-1}(\sigma)}{2}+\frac{\delta}{2}}{0.366519} \quad (19)$$

4.4. Tahap Pengembangan

4.4.1. Tahap Dasar

Dalam penelitian ini, dibutuhkan instalasi TORCS yang telah disematkan dengan *patch* Simulated Car Racing Championship Server. Pengendali yang menjadi bahan pengembangan pengendali dalam penelitian ini tersedia dalam package SCR Client. Paket ini dapat diekstrak menuju lokasi sesuai preferensi dan dapat digunakan dalam IDE yang digunakan. Untuk pengerjaan pengendali baru, kelas SimpleDriver dapat di salin dengan nama pengendali yang akan dikembangkan, yang mana pada penelitian ini kelas yang digunakan bernama KalmanDriver.

4.4.2. Implementasi Fungsi Kalman Filter

Pengaplikasian Kalman Filter diimplementasikan sebagai kelas baru dengan masing-masing variabel berbentuk *array* dan menggunakan konstan `stepTime` yang merupakan waktu *step* simulasi dalam permainan. Variabel $\hat{x}_0$ dan P_0 untuk model filter diinisialisasikan dengan nilai:

$$\phi = -\frac{\frac{\tan^{-1}(\sigma)}{2}+\frac{\delta}{2}}{0.366519} \quad (20a)$$

$$\phi = -\frac{\frac{\tan^{-1}(\sigma)}{2}+\frac{\delta}{2}}{0.366519} \quad (20b)$$

Untuk matriks derau Q dan R masing-masing digunakan nilai:

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \quad (21a)$$

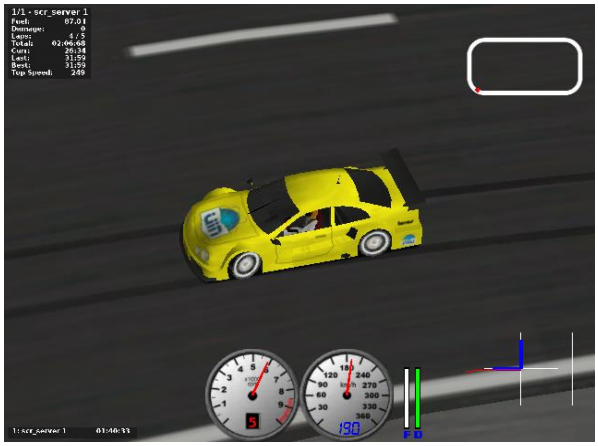
$$R = \begin{bmatrix} 15000 & 0 & 0 & 0 \\ 0 & 15000 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (21b)$$

4.4.3. Implementasi Fungsi Lainnya

Untuk tahap pengerjaan fungsi lainnya, penulis mengimplementasikan permodelan ketiga fungsi dasar pengendali yang telah dijelaskan pada poin-poin sebelumnya. Untuk fungsi pergantian gigi, penulis mengubah konstan `gearUp` dan `gearDown` pada pengendali yang dikembangkan, sesuai dengan konstan nilai maksimal dan minimal yang telah dijabarkan pada Tabel 1 dengan asumsi fungsi untuk pergantian gigi sudah tepat dan tidak membutuhkan perubahan.
Untuk fungsi pengendalian kecepatan, fungsi `getRadius` dan `getRealRadius` ditambahkan pada pengendali. Fungsi `getRadius` mengimplementasikan proses perhitungan radius tikungan menggunakan sensor lintasan, sementara `getRealRadius` mengimplementasikan penentuan perhitungan berdasarkan sensor lintasan. Hasil perhitungan radius lintasan kemudian disematkan pada perhitungan kecepatan target pada fungsi `getAccel`.
Terakhir, fungsi kemudi dalam pengendali diselesaikan pada fungsi `getSteer` dimana perhitungan target jarak, posisi x serta y menggunakan sensor lintasan, dan disematkan sebagai bahan pengukuran dalam fungsi Kalman Filter pada kelas terpisah.

4.5. Tahap Pengujian

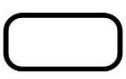
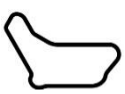
Pengujian untuk pengendali KalmanDriver yang telah dikembangkan dilakukan pada lima sirkuit yang tersedia dalam instalasi TORCS, diantaranya a-speedway, e-track-5, g-track-1, g-track-2, dan g-track-3. Dalam pengujian ini, pengendali menjalankan tiap sirkuit selama lima putaran untuk mencoba mencetak waktu tercepat dan kecepatan tertinggi dalam tiap sirkuit, dan hasilnya akan dibandingkan dengan pengendali Ground Truth Driver yang merupakan pengendali sama tanpa menggunakan pengaplikasian Kalman filter. Ground Truth Driver dijalankan pada dua pengaturan berbeda, yaitu pengaturan normal dan pengaturan berderau (melalui *command line* noisy), sementara pengendali KalmanDriver dijalankan dengan pengaturan noisy saja.



Gambar 6. Tampak mobil yang digunakan dalam pengujian pengendali dalam permainan.

Mobil yang digunakan dalam pengujian ini adalah car5-trb1 yang merupakan bagian dari instalasi TORCS, dengan berat 1150 kg dan penggerak roda belakang, tenaga 395kW yang setara dengan 580 tenaga kuda, serta area *coefficient drag* 0,69 m^2 [2], menjadikannya salah satu mobil yang kompetitif dengan mobil lainnya dalam kategori trb1 dengan keunggulan aerodinamikanya.

Tabel 3. Daftar Sirkuit yang Digunakan dalam Pengujian Beserta Informasi Lengkap Mereka.

Ilustrasi					
Nama	a-speedway	g-track-1	e-track-5	g-track-2	g-track-3
Panjang	1908,32 m	2507,56 m	1621,73 m	3185,83 m	2843,10 m
Lebar	25,00 m	15,00 m	20,00 m	15,00 m	10,00 m

Dalam pengujian, KalmanDriver mencetak waktu yang lebih cepat dari Ground Truth Driver pada dua sirkuit dalam pengaturan yang sama (a-speedway dan g-track-1) sementara pada sirkuit lainnya KalmanDriver lebih lambat dengan selisih kurang dari satu detik.

Akan tetapi, pergerakan dari KalmanDriver lebih halus dan raihan kecepatan tinggi pada lintasan lurus sedikit lebih besar pada ketiga lintasan dibandingkan GTD, sementara GTD memiliki racing line yang lebih baik dari KalmanDriver.

Masalah pergoyangan mobil terjadi beberapa kali dalam pengujian, beberapa kasus yang penting terjadi pada a-speedway Ground Truth Driver, pada pengaturan noisy dimana pergerakannya menyebabkan mobil menabrak dinding pembatas saat melintir. Hal yang sama terjadi pada GTD dengan pengaturan normal pada g-track-1, dan g-track-3 dimana pendaratan yang tidak sempurna menyebabkan mobil melintir pada seluruh pengendali. Dalam pengujian g-track-3, GTD pada pengaturan normal tidak bisa melanjutkan pengujian setelah tiga putaran setelah mencapai poin kerusakan maksimal, sementara kedua pengendali dalam pengaturan noisy dapat menyelesaikan kelima putaran meskipun dengan pendaratan tidak sempurna dan melintir pada lintasan bergelombang dikarenakan pergerakan mereka yang sedikit bergoyang.

Tabel 4. Hasil Lengkap Raihan Waktu Tercepat dan Kecepatan Tertinggi dalam Pengujian Kalmandriver dan Groundtruthdriver dalam Lima Sirkuit Berbeda

Sirkuit Pengujian	GTD	GTD (noisy)	KalmanDriver (noisy)
a-speedway	31,59 detik 249 kpj	42,07 detik 220 kpj	35,62 detik 236 kpj
g-track-1	43,29 detik 225 kpj	44,71 detik 225 kpj	44,80 detik 224 kpj
e-track-5	28,71 detik 223 kpj	34,09 detik 204 kpj	34,27 detik 203 kpj
g-track-2	58,07 detik 251 kpj	64,45 detik 242 kpj	63,92 detik 243 kpj
g-track-3	68,06 detik 218 kpj	71,76 detik 214 kpj	72,75 detik 213 kpj

Dari seluruh hasil pengujian, bisa disimpulkan bahwa Kalman Driver memiliki pergerakan yang halus dengan menggunakan Kalman Filter untuk mengestimasikan pergerakan selanjutnya dan lebih cepat pada beberapa lintasan. Akan tetapi, pergoyangan kiri dan kanan kemungkinan beresiko pada beberapa lintasan, seperti contoh e-track-3, dimana pergerakannya dapat memelintirkan mobil yang dikendalikan baik saat mendarat dari lompatan atau melintasi permukaan lintasan yang bergelombang.

5. KESIMPULAN DAN SARAN

Dalam penelitian ini, penulis menyimpulkan bahwa pengendali untuk TORCS dikembangkan berdasarkan kelas SimpleDriver yang disediakan oleh paket *client* dari SCR Server. Kelas ini terdiri dari pengaplikasian fungsi aktuator untuk tingkatan input kemudi, akselerasi, rem, dan lainnya dalam permainan menggunakan algoritma yang digunakan serta permodelan sensor yang ada dalam paket tersebut.

Penggunaan Kalman Filter diaplikasikan dalam pengendali yang dikembangkan dalam pergerakan mobil melalui kemudi dimana prediksi dan koreksi pergerakan dilakukan berdasarkan titik target vektor sebagai bahan pengukuran untuk sistem filter yang diselesaikan pada kelas yang terpisah. Derau untuk pengukuran serta proses juga dimodelkan untuk mengatasi derau dalam permainan.

Dalam pengujian yang dilakukan, pengendali Kalman Driver memiliki performa yang baik dalam menghaluskan pergerakan bahkan dalam pengaturan *noisy* dibandingkan dengan pengendali Ground Truth Driver yang merupakan versi dasar dari Kalman Driver. Akan tetapi gangguan pergerakannya dapat menyebabkan mobil melintir, terutama saat melintasi lintasan dengan permukaan yang tidak rata.

Penulis menyadari ada beberapa kekurangan yang ada dalam penelitian ini. Diharapkan bahwa pengaplikasian Kalman Filter dalam pengendali ini dapat dioptimalkan lebih baik, baik dengan permodelan yang lebih dalam, penggunaan implementasi Kalman Filter lainnya (seperti Extended Kalman Filter), maupun dipadukan dengan algoritma lainnya (seperti *fuzzy logic*, *neural network*, dan lainnya). Selain itu, penulis berharap *The Open Racing Car Simulator* (TORCS) dapat digunakan dalam aktivitas perkuliahan sebagai sarana pembelajaran dalam bidang kecerdasan buatan dan menjadi referensi untuk pengembangan pengendali berbasis kecerdasan buatan, serta pengendali dapat dibuat menggunakan algoritma lainnya untuk penelitian selanjutnya dengan topik serupa.

DAFTAR PUSTAKA

[1] B. Wymann. "Welcome to TORCS Racing Board." Internet: http://www.berniw.org/trb/index.php, Dec. 1, 2004 [Jul. 24, 2017].

[2] B. Wymann. "car5-trb1." Internet: http://www.berniw.org/trb/cars/car_view.php?viewcarid=3, Jun. 15, 2006 [Apr. 10, 2018].

[3] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, A. Sumner. "TORCS: The Open Racing Car Simulator, v1.3.7." 2015

[4] D. Loiacono, et al. "The 2009 Simulated Car Racing Championship." *IEEE Transactions on Computational Intelligence and AI in Games*, Volume 2, May 2010, Pages 131-147, https://doi.org/10.1109/TCIAIG.2010.2050590

[5] D. Loiacono, L. Cardamone, and P. L. Manzi. "Simulated Car Racing Championship Competition Software Manual." *arXiv preprint*, April 2013, Pages 1304-1672, https://arxiv.org/abs/1304.1672v2.

[6] I. Millington and J. Funge. *Artificial Intelligence for Games. (2nd edition)*. Boca Raton, FL: CRC Press, 2009, pp. 7-8.

[7] J. Gomez-Gil, R. Ruiz-Gonzalez, S. Alonso-Garcia, and F. J. Gomez-Gil. "A kalman filter implementation for precision improvement in low-cost GPS positioning of tractors." *Sensors*, Volume 13, November 2013, pp. 15307-15323, https://doi.org/10.3390/s131115307.
[8] J. Koutnik, J. Schmidhuber, and F. Gomez. "Online evolution of deep convolutional network for vision-based reinforcement learning." *International Conference on Simulation of Adaptive Behavior*, Jul. 2014, Pages 260-269, https://doi.org/10.1007/978-3-319-08864-8_25.
[9] J. Wexler. (May 2002). *Artificial Intelligence in Games: A look at the smarts behind Lionhead Studio's "Black and White" and where it can and will go in the future*. [Online]. Available: https://www.academia.edu/31386126/Artificial_Intelligence_in_Games_A_look_at_the_smarts_behind_Lionhead_Studios_Black_and_White_and_where_it_can_and_will_go_in_the_future
[10] K. Albelihi. "The Gazelle Adaptive Racing Car Pilot." M.A. thesis, Indiana University South Bend, USA, 2014.
[11] K. J. Kim, J. H. Seo, J. G. Park, and J. C. Na. "Generalization of TORCS Car Racing Controllers with Artificial Neural Networks and Linear Regression Analysis." *Neurocomputing*, Volume 88, July 2012, Pages 87-99, https://doi.org/10.1016/j.neucom.2011.06.034.
[12] L. C. Hun, O. L. Yeng, L. T. Sze, and K. V. Chet. (2016, June 8). *Kalman Filtering and Its Real-Time Applications, Real-time Systems*. [On-line]. 17(1). Available: https://www.intechopen.com/download/pdf/50419 [September 12, 2018].
[13] L. Cardamone, P. L. Manzi, D. Loiacono, and E. Onieva. "Advanced overtaking behaviors for blocking opponents in racing games using a fuzzy architecture." *Expert Systems with Applications*, Volume 40, November 2013, Pages 6447-6458, https://doi.org/10.1016/j.eswa.2013.04.030.
[14] M. B. Rhudy, R. A. Salguero, and K. Holappa. "A Kalman Filtering Tutorial for Undergraduate Students." *International Journal of Computer Science & Engineering Survey (IJCSES)*, Volume 8, February 2017, Pages 1-18, https://doi.org/10.5121/ijcses.2017.8101.
[15] M. R. Bonyadi, Z. Michalewicz, S. Nallaperuma, and F. Neumann. "Ahura: A heuristic-based racer for the open racing car simulator." *IEEE Transactions on Computational Intelligence and AI In Games*, Volume 9, May 2016, pp. 290-304, https://doi.org/10.1109/TCIAIG.2016.2565661.
[16] M. Salem, A.M Mora, J.J. Merello, dan P. García-Sánchez. "Driving in TORCS using Modular Fuzzy Controllers." *European Conference on the Applications of Evolutionary Computation*, Volume 1, March 2017, Pages 361-376, https://doi.org/10.1007/978-3-319-55849-3_24.
[17] M. T. Chan, C. W. Chan, and C. Gelowitz. "Development of a Car Racing Simulator Game Using Artificial Intelligence Techniques." *International Journal of Computer Games Technology*, Volume 2015, November 2015, Pages 12-17. https://doi.org/10.1155/2015/839721.
[18] MathWorks Inc. "Using Kalman Filter for Object Tracking." Internet: https://www.mathworks.com/help/vision/examples/using-kalman-filter-for-object-tracking.html, Sep. 14, 2012 [Oct. 18, 2017]
[19] Mmervine. "Pole Position." Internet: https://retrogameguy.com/2013/05/12/pole-position/, May 12, 2013 [Aug. 27, 2017]
[20] N. Ireson. "GT5 Eat Your Heart Out: Video Game Moment Of Zen." Internet: https://www.motorauthority.com/news/1066384_gt5-eat-your-heart-out-video-game-moment-of-zen, Sep. 20, 2011 [Nov. 11, 2018]
[21] N. van Hoorn, J. Togelius, D. Wierstra, and J. Schmidhuber. "Robust player imitation using multiobjective evolution." *2009 IEEE Congress on Evolutionary Computation*, May 2009, Pages 652-659, https://doi.org/10.1109/CEC.2009.4983007.
[22] R. Faragher. "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation." *IEEE Signal Processing Magazine*, Volume 29, August 2012, Pages 128-132, https://doi.org/10.1109/MSP.2012.2203621.
[23] R. Rudzits and N. Pugeault. "Efficient learning of pre-attentive steering in a driving school framework." *KI-Künstliche Intelligenz*, Volume 29, February 2015, Pages 51-57, https://doi.org/10.1007/s13218-014-0340-1.
[24] S. Nallaperuma, F. Neumann, M. R. Bonyadi, and Z. Michalewicz. "EVOR: an online evolutionary algorithm for car racing games." *Genetic and evolutionary computation*, Volume #, July 2014, Pages 317–324, https://doi.org/10.1145/2576768.2598298.
[25] S. Sanchez and S. Cussat-Blanc. "Gene regulated car driving : using a gene regulatory network to drive a virtual car." *Genetic Programming and Evolvable Machines*, Volume 15, June 2014, Pages 477-511, http://dx.doi.org/10.1007/s10710-014-9228-y
[26] SourceForge. "TORCS - The Open Racing Car Simulator." Internet: https://sourceforge.net/projects/torcs/, Apr. 16, 2019 [Aug. 27, 2017]
[27] Suyanto. *Artificial Intelligence – Searching, Reasoning, Planning, dan Learning*. Bandung: Informatika, 2014, pp. 3-11.
[28] V. Amrizal and Q. Aini. *Kecerdasan Buatan*. Jakarta: Halaman Moeka, 2013, pp. 1-2.

NOMENKLATUR

$\hat{x}$	Matriks vektor waktu sistem
x	Posisi target berdasarkan koordinat x sesuai arah mobil
y	Posisi target berdasarkan koordinat y sesuai arah mobil
ρ	Jarak menuju posisi target yang dituju
θ	Sudut dari titik target yang dituju
ϕ	Input kemudi untuk mobil
ts_n	Keluaran sensor lintasan ke-n
rad_n	Sudut dari sensor lintasan ke-n dalam satuan radian
δ	Sudut arah mobil terhadap sumbu lintasan
P	Matriks kovarien error sistem
B_{k-1}	Matriks pengaruh luar sistem
μ	Matriks vektor kendali
ν	Matriks derau proses dalam prediksi.
z_k	Matriks input pengukuran
H	Matriks observasi model Kalman Filter

I	Matriks identitas
w	Matriks derau pengukuran; berasal dari sensor lintasan
F_{k-1}	Matriks prediksi sistem pada waktu k sebelumnya
Δt	Konstan waktu simulasi server (0.02 detik / 20 milidetik)
Q	Matriks derau kovarien
Y_k	Matriks error pada waktu k
z_k	Matriks pengukuran pada waktu k
K_k	Kalman gain pada waktu k
R_k	Matriks derau proses pada waktu k
s	Semiperimeter segitiga
A	Luas dari segitiga yang terdiri dari panjang sisi dan semiperimeter
r	Radius tikungan yang terbentuk berdasarkan luas segitiga A
v'	Kecepatan target pengendali
a	Input akselerasi dan rem pada pengendali