p-ISSN 2301–4156 | e-ISSN 2460–5719

Kendali *Inverted Pendulum*: Studi Perbandingan dari Kendali Konvensional ke *Reinforcement Learning*

Ahmad Ataka[1], Andreas Sandiwan[2], Hilton Tnunay[3], Dzuhri Radityo Utomo[4], Adha Imam Cahyadi[5]

[1,2,4,5] Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada, Jl. Grafika No. 2 Kampus UGM, Yogyakarta 55281 INDONESIA (tel.: 0274-552305; email:[1]ahmad.ataka.ar@ugm.ac.id, [2]andreassandiwan93@mail.ugm.ac.id, [4]dzuhri.r.u@ugm.ac.id, [5]adha.imam@ugm.ac.id)
[3] Geomatics Section, Faculty of Engineering Technology, KU Leuven, 9000 Ghent, BELGIA (email: [3]hilton.tnunay@kuleuven.be)

[Diterima: 9 Mei 2023, Revisi: 17 Juli 2023]
Corresponding Author: Ahmad Ataka

INTISARI — Perkembangan dalam *deep reinforcement learning* dalam beberapa tahun terakhir telah mendorong penggunaannya dalam memecahkan masalah yang menantang, seperti permainan catur dan Go. Namun, muncul pertanyaan tentang cocok tidaknya kelas metode ini untuk memecahkan masalah kendali secara umum, seperti mobil tanpa pengemudi, kendali robot bergerak, atau kendali manipulator industri. Dalam makalah ini, disajikan studi perbandingan antara berbagai kelas algoritma kendali dan *reinforcement learning* dalam mengendalikan sistem *inverted pendulum*. Kinerja kendali berbasis *root locus*, kendali *state compensator*, kendali proporsional-derivatif (PD), dan metode *reinforcement learning*, seperti *proximal policy optimization* (PPO), diuji untuk mengontrol *inverted pendulum* pada sebuah gerobak. Kinerja dievaluasi berdasarkan respons transien (seperti *overshoot*, *peak time*, dan *settling time*) dan respons kondisi tunak (kesalahan kondisi tunak dan energi total). Hasil penelitian menunjukkan bahwa algoritma *reinforcement learning* dapat menghasilkan solusi yang setara dengan algoritma kendali, meskipun tanpa mengetahui properti sistem. Oleh karena itu, algoritma ini paling cocok digunakan untuk mengendalikan *plant* tanpa banyak informasi mengenai model, sehingga pengujian aturan tertentu lebih mudah dan aman. Penggunaan algoritma ini juga dianjurkan untuk sistem dengan fungsi objektif yang jelas.

KATA KUNCI — *Reinforcement Learning*, *Inverted Pendulum*, *Root Locus*, Umpan Balik Keadaan, Kendali PD.

I. PENDAHULUAN

Bidang *reinforcement learning* telah mendapatkan perhatian luas dalam beberapa tahun terakhir. Meskipun perkembangannya dimulai pada tahun 1990-an [1], popularitasnya baru-baru ini meningkat dengan pesat berkat kemajuan dalam *deep learning* pada tahun 2012 [2]. *Deep reinforcement learning* muncul dari penggabungan *deep neural network*, yang mampu melakukan pendekatan terhadap fungsi kompleks, dengan kerangka kerja *reinforcement learning* yang dapat menyelesaikan masalah kendali tertentu. Masalah kendali ini melibatkan pengambilan keputusan atau tindakan untuk sistem tertentu (atau lingkungan). Keberhasilan besar *deep reinforcement learning* dalam memecahkan permainan, seperti Atari [3], [4], telah mendorong aplikasinya untuk menyelesaikan tantangan dalam permainan lain, seperti catur, Go, dan Dota2 [5]-[7].

Di antara masalah yang hendak dipecahkan oleh *deep reinforcement learning* adalah masalah kendali dan robotika tradisional seperti kendali mobil tanpa pengemudi [8], navigasi robot bergerak [9], kendali robot berkaki [10], atau kendali manipulator industri [11]. Meskipun telah berhasil, masih ada tantangan dalam menerapkan *deep reinforcement learning* untuk masalah kendali umum dibandingkan dengan masalah permainan.

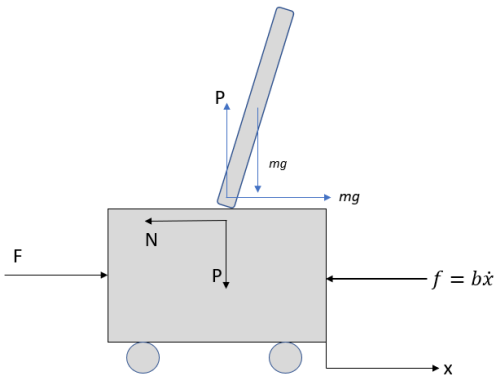
Salah satu tantangan utama adalah kurangnya fungsi penghargaan yang jelas [12]. Meskipun lingkungan permainan biasanya memberikan kriteria keberhasilan yang jelas (seperti sistem penilaian atau kondisi menang), umumnya tidak demikian dalam masalah kendali. Seperti yang telah dipelajari secara ekstensif, pilihan *reward* sangat penting untuk kinerja *reinforcement learning* yang mendalam [12]. Oleh karena itu, penyelesaian masalah kendali umum menggunakan *reinforcement learning* menciptakan masalah baru dalam merancang fungsi *reward* yang mencerminkan kriteria keberhasilan masalah kendali. Tantangan lain dalam penerapan *reinforcement learning* untuk masalah kendali adalah sifat nonepisodik dari masalah tersebut [13].

Terlepas dari masalah yang disebutkan di atas, ada juga tantangan dalam melakukan pelatihan dalam masalah kendali [12]. Meskipun sangat mudah melakukan pelatihan untuk masalah permainan, melatih sistem kendali nyata secara langsung terbukti rumit. Karena *reinforcement learning* membutuhkan sejumlah besar data pelatihan untuk menemukan aturan yang diinginkan, tidak ada jaminan bahwa aturan yang diterapkan pada suatu saat tertentu tidak mengarah pada hasil yang buruk. Banyak peneliti mencoba mengatasi masalah ini dengan melatih aturan dalam simulasi sebelum menerapkan aturan tersebut dalam sistem nyata, sebuah teknik yang disebut *sim-to-real* [12]. Hal ini sangat berhasil dalam kasus-kasus yang kompleks, seperti melatih robot berkaki empat untuk melakukan tugas berlari. Namun, metode ini bergantung pada ketersediaan simulator berbasis fisika yang harus cukup canggih untuk meniru kinerja sistem nyata.

Ada juga beberapa tulisan yang melaporkan fenomena kinerja yang tidak stabil dari *deep reinforcement learning* ketika diterapkan pada beberapa masalah kendali [12]. Hal ini berbeda dengan metode kendali konvensional. Pada metode konvensional, kestabilan biasanya dapat dianalisis dan dijamin jika memungkinkan. Selain itu, pada sebagian besar masalah kendali, tujuannya bukan hanya "menyelesaikan masalah" itu sendiri, tetapi juga menghasilkan kinerja dengan spesifikasi tertentu.

Untuk mengevaluasi kinerja *reinforcement learning* dalam masalah kendali, dalam makalah ini, dilakukan studi perbandingan antara algoritma *reinforcement learning*, yaitu *proximal policy optimization* (PPO), dan beberapa algoritma kendali konvensional (*compensator* berbasis *root locus*, *state compensator*, dan kendali proporsional-derivatif/PD) untuk kendali *inverted pendulum*. Masalah pengendalian *inverted pendulum* dipilih karena merupakan salah satu masalah standar

p-ISSN 2301–4156 | e-ISSN 2460–5719



Gambar 1. *Free-body diagram* dari *inverted pendulum* pada gerobak, juga dikenal sebagai *cartpole*.

yang digunakan untuk menguji berbagai algoritma kendali [14]. Hal ini terjadi karena *inverted pendulum* merupakan sistem nonlinear dengan kesetimbangan yang tidak stabil, tetapi pada saat yang sama dapat dengan mudah dilinearisasi pada lokalitas kesetimbangannya. Ditemukan bahwa dengan jumlah pelatihan yang cukup, algoritma PPO dapat mencapai kinerja transien dan kondisi tunak yang sebanding dibandingkan dengan metode kendali konvensional, meskipun pengetahuan tentang model pendulum tidak dimiliki. Hasil penelitian ini menegaskan bahwa *reinforcement learning* memiliki potensi yang sangat besar untuk menghasilkan kinerja yang memenuhi spesifikasi tertentu yang mirip dengan algoritma kendali konvensional.

Pada Bagian II, disajikan model matematis dari *inverted pendulum*, termasuk strategi untuk memodifikasi masalah kendali ke dalam formulasi *reinforcement learning*, dilanjutkan dengan deskripsi algoritma kendali pada Bagian III, dan diikuti dengan deskripsi mengenai skenario pengujian pada Bagian IV. Hasil dan diskusi disajikan pada Bagian V, kesimpulan dan pekerjaan di masa depan disajikan pada Bagian VI.

II. MODEL INVERTED PENDULUM

A. DERIVASI MODEL

Inverted pendulum pada sistem gerobak ditunjukkan pada Gambar 1. Masukan yang diberikan ke sistem adalah gaya horizontal $u = F$. Keadaan sistem dituliskan sebagai berikut.

$$x_s = \begin{bmatrix} x_c & \dot{x}_c & \theta & \dot{\theta} \end{bmatrix}^T \quad (1)$$

dengan x_c mengacu pada perpindahan gerobak, sedangkan θ mengacu pada sudut pendulum terhadap sumbu vertikal.

Keluaran sistem akan bergantung pada sensor yang tersedia. Untuk beberapa pengendali yang dianalisis dalam makalah ini, diasumsikan semua informasi status diketahui. Dalam hal ini,

$$y = x_s. \quad (2)$$

Sementara itu, yang lainnya hanya memerlukan parameter utama untuk dikendalikan, yaitu sudut *inverted pendulum* θ, yang dalam kasus ini adalah

$$y = \theta. \quad (3)$$

Untuk memodelkan *inverted pendulum*, dianalisis dinamikanya menggunakan hukum Newton II. Untuk menyederhanakan analisis, sistem dibagi menjadi dua komponen, yaitu pendulum dan kereta. Dengan mengasumsikan lingkungan tanpa gesekan, dinamika gerobak dalam arah horizontal dapat dituliskan (4).

$$F - N = m_c \ddot{x}_c \quad (4)$$

dengan N mengacu pada gaya kontak horizontal antara pendulum dan gerobak, sedangkan m_c mengacu pada massa kereta.

Dinamika pendulum pada arah horizontal dapat dituliskan sebagai berikut.

$$N = m_p \ddot{x}_p. \quad (5)$$

Akselerasi pendulum di sini adalah kombinasi dari akselerasi kereta $\ddot{x}_c$ dan rotasi tiang. Misalnya, tiang memiliki kecepatan sudut $\dot{\theta}$ dan percepatan sudut $\ddot{\theta}$. Gabungan percepatan gerobak dan gerakan pendulum yang berputar menyebabkan percepatan dalam arah horizontal seperti pada (6).

$$\ddot{x}_p = \ddot{x}_c + \ddot{\theta} l \cos\theta - \dot{\theta}^2 l \sin\theta. \quad (6)$$

Selanjutnya, dengan menganalisis dinamika rotasi pendulum, didapatkan persamaan berikut.

$$\tau = I\ddot{\theta} + \tau_{inertial}. \quad (7)$$

Di sini, $\tau_{inertial}$ mengacu pada torsi yang dihasilkan oleh gaya inersia akibat akselerasi kereta. Dengan memasukkan momen inersia pendulum $I = \frac{4}{3} m_p l^2$ dan torsi karena berat pendulum $\tau = m_p g l \sin\theta$, persamaan tersebut dapat ditulis ulang sebagai (8)

$$m_p g l \sin\theta = \frac{4}{3} m_p l^2 \ddot{\theta} + m_p \ddot{x}_c l \cos\theta. \quad (8)$$

Dengan menggabungkan tiga persamaan dinamika, diperoleh model nonlinear dari *inverted pendulum* sebagai berikut.

$$F = (m_c + m_p)\ddot{x}_c + m_p(\ddot{\theta} l \cos\theta - \dot{\theta}^2 l \sin\theta) \quad (9)$$

$$m_p g l \sin\theta = \frac{4}{3} m_p l^2 \ddot{\theta} + m_p \ddot{x}_c l \cos\theta \quad (10)$$

B. LINEARISASI

Untuk mengimplementasikan kendali linear, model *inverted pendulum* perlu dilinearisasi di dekat titik ekuilibriumnya ($\theta = 0$). Linearisasi sederhana dapat dilakukan hanya dengan modifikasi berikut.

$$\sin\theta \approx \theta \quad (11)$$

$$\cos\theta \approx 1 \quad (12)$$

$$\dot{\theta}^2 \approx 0. \quad (13)$$

Oleh karena itu, model dapat ditulis dalam bentuk berikut.

$$F = (m_c + m_p)\ddot{x}_c + m_p l \ddot{\theta} \quad (14)$$

$$m_p g l\theta = \frac{4}{3} m_p l^2 \ddot{\theta} + m_p l \ddot{x}_c \quad (15)$$

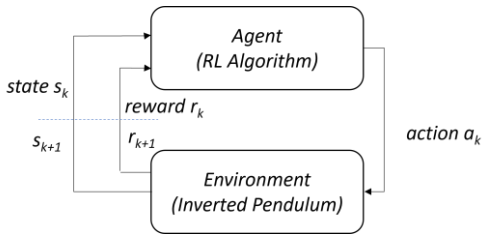
yang dapat ditulis sebagai

$$F = (m_c + m_p)\ddot{x}_c + m_p l \frac{(m_p g l\theta - m_p l \ddot{x}_c)}{\frac{4}{3} m_p l^2} \quad (16)$$

$$F = \left(m_c + \frac{m_p}{4}\right)\ddot{x}_c + \frac{3}{4} m_p g\theta \quad (17)$$

dan

$$m_p g l\theta = \frac{4}{3} m_p l^2 \ddot{\theta} + \frac{m_p l(F - m_p l\ddot{\theta})}{(m_c + m_p)} \quad (18)$$



Gambar 2. Kendali *inverted pendulum* sebagai masalah *reinforcement learning*.

$$g\theta = \left(\frac{4}{3} - \frac{m_p}{m_c+m_p}\right) l\ddot{\theta} + \frac{1}{(m_c+m_p)} F \quad (19)$$

$$g\theta = \left(\frac{4m_c+3m_p}{3(m_c+m_p)}\right) l\ddot{\theta} + \frac{1}{(m_c+m_p)} F. \quad (20)$$

Kemudian, persamaan tersebut dapat dimodifikasi menjadi persamaan keadaan sebagai berikut.

$$\dot{x}_s = Ax_s + Bu \quad (21)$$

dengan

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & b & 0 \end{bmatrix} \quad (22)$$

$$a = -\frac{3m_p g}{4m_c+m_p} \quad (23)$$

$$b = \frac{3(m_c+m_p)}{4m_c+3m_p}\frac{g}{l} \quad (24)$$

$$B = \left[0 \quad \frac{1}{m_c+\frac{m_p}{4}} \quad 0 \quad -\frac{3}{l(4m_c+3m_p)}\right]^T. \quad (25)$$

Sistem yang dilinearisasi juga dapat ditransformasikan ke dalam domain frekuensi yang kompleks.

$$F(s) = (m_c + m_p)s^2 X_c(s) + m_p l s^2 \theta(s) \quad (26)$$

$$m_p g l\theta(s) = \frac{4}{3} m_p l^2 s^2 \theta(s) + m_p l s^2 X_c(s). \quad (27)$$

Jadi, fungsi alih dapat dituliskan sebagai berikut.

$$G(s) = \frac{\theta(s)}{F(s)} = \frac{c}{s^2-d} \quad (28)$$

dengan $c = \frac{d}{(m_p+m_c)g}$ dan $d = \frac{g}{l\left(\frac{4}{3}-\frac{m_p}{m_p+m_c}\right)}$.

Dari fungsi alih, dapat dilihat bahwa *pole* sistem yang dilinearisasi terletak pada

$$s_1 = \sqrt{d},\ s_2 = -\sqrt{d}. \quad (29)$$

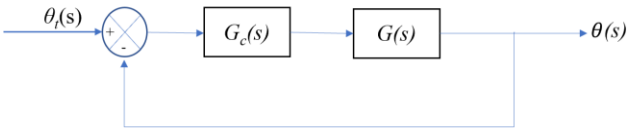
Pole pertama inilah yang membuat sistem tidak stabil.

C. INVERTED PENDULUM SEBAGAI MASALAH REINFORCEMENT LEARNING

Masalah *reinforcement learning* berkaitan dengan sistem dengan dinamika pembaruan keadaan yang tidak diketahui $s_{k+1} = F(s_k, a_k)$. Di sini, s_k dan a_k mengacu pada keadaan sistem dan aksi pada *iterasi* ke-*k*. Selain itu, sistem juga akan mengembalikan nilai *reward* r_k. Masalah *reinforcement learning* bertujuan untuk menemukan himpunan aksi a_k yang memaksimalkan keuntungan yang diharapkan selama *M* langkah, yang didefinisikan sebagai (30).

$$R = \sum_{k=0}^{M} r_k. \quad (30)$$

Kendali *inverted pendulum* dapat dilihat sebagai masalah *reinforcement learning* seperti yang disajikan pada Gambar 2.



Gambar 3. Diagram blok *compensator* berbasis *root locus*.

Di sini, keadaan sistem s_k dari perspektif masalah *reinforcement learning* sama dengan keadaan x_s dari perspektif kendali modern, meskipun dalam bentuk diskret. Demikian pula, aksi a_k adalah sinyal kendali u. Satu-satunya parameter yang tersisa adalah fungsi *reward* yang harus dirancang sedemikian rupa sehingga merepresentasikan kinerja sistem *inverted pendulum* dalam mencoba untuk tetap berada di posisi vertikal. Dalam hal ini, digunakan fungsi *reward* yang diterapkan oleh OpenAI di lingkungan CartPole Gym. *Reward*-nya $r_k = 1$ jika posisi *cartpole* memenuhi $-x_{th} < x_c < x_{th}$ dan sudut *cartpole* memenuhi $-\theta_{th} < \theta < \theta_{th}$, dengan x_{th} dan θ_{th} merepresentasikan ambang batas posisi dan sudut. Kemudian, jumlah langkah maksimum *M* dipilih. Dengan kata lain, algoritma *reinforcement learning* yang baik akan menghasilkan tindakan a_k lebih dari *M* langkah yang mengarah ke pengembalian maksimum *M* r_k.

III. ALGORITMA KENDALI

Ada empat algoritma kendali yang dibahas dalam makalah ini. Tiga metode pertama dipilih karena banyak digunakan dalam literatur kendali, terutama untuk sistem linear atau terlinearisasi. Satu metode dipilih berdasarkan pendekatan kendali klasik (menggunakan fungsi alih), yaitu metode *root locus*, dan satu metode lainnya berdasarkan pendekatan kendali modern (menggunakan persamaan keadaan), yaitu *state compensator*. Metode kendali ketiga adalah kendali PD, yang dipilih karena penggunaannya yang populer di industri. Terakhir, dipilih algoritma PPO sebagai salah satu algoritma *reinforcement learning* yang canggih untuk sistem dengan aksi kontinu.

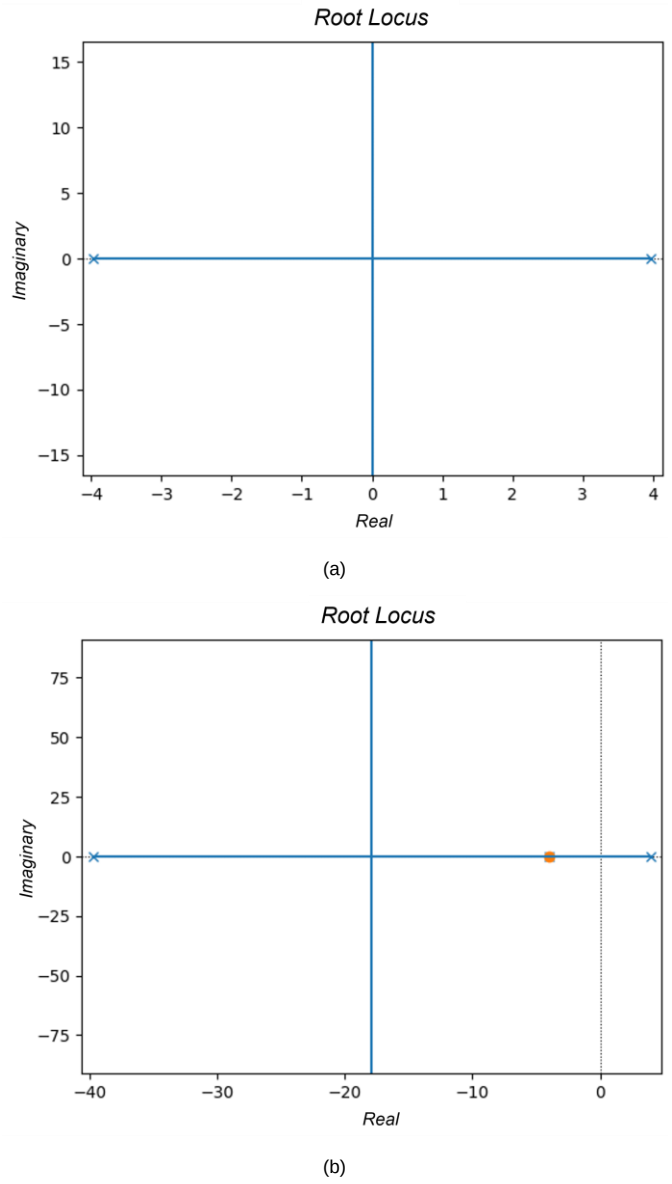
A. COMPENSATOR BERBASIS ROOT LOCUS

Dalam metode ini, akan ditambahkan *compensator* untuk memodifikasi *root locus* sistem sedemikian rupa sehingga *root locus* yang baru akan melewati *pole* yang diinginkan berdasarkan spesifikasi kebutuhan. *Compensator* $G_c(s)$ akan diberi masukan berupa kesalahan antara keluaran target dan keluaran yang sesungguhnya, $e = \theta_t - \theta$, dan menghasilkan sinyal kendali u ke dalam sistem. Diagram blok pengendali yang diusulkan ditunjukkan pada Gambar 3.

Sebelum mendesain *compensator*, diperlukan analisis *root locus* dari fungsi alih pendulum. *Root locus* dari sistem *inverted pendulum* yang dilinearisasi dapat diamati pada Gambar 4(a). Dari *root locus*, dapat dilihat bahwa pengendali penguatan sederhana tidak akan membuat *inverted pendulum* stabil karena akan selalu ada setidaknya satu *pole* sistem yang terletak di bagian kanan bidang frekuensi kompleks atau tepat pada sumbu imajiner.

Untuk menstabilkan sistem, dapat ditambahkan *zero* dari *compensator* di tempat *pole* s_2, yaitu $z_{comp} = s_2$. Kemudian, untuk menarik *root locus* ke arah setengah bidang kiri, ditambahkan *pole* dari *compensator* ke lokasi $s_{comp} = 10s_2$. Metode ini lebih efektif daripada mencoba untuk mengeliminasi *pole* positif s_1 secara langsung karena pada kenyataannya *pole* positif tidak akan benar-benar dieliminasi,

p-ISSN 2301–4156 | e-ISSN 2460–5719



Gambar 4. *Root locus*, (a) *root locus* dari *inverted pendulum*, (b) *root locus* dari sistem yang dimodifikasi.

yang menyebabkan sistem tetap tidak stabil. *Root locus* yang baru ditunjukkan pada Gambar 4(b).

Selanjutnya, dipilih penguatan *compensator* menjadi $K_{comp} = 300$ untuk memastikan bahwa kedua *pole* kalang tertutup yang baru dari sistem terletak di setengah bidang yang kiri, seperti ditunjukkan pada Gambar 5. Fungsi alih akhir dari *compensator* dituliskan dalam (31).

$$G_c(s) = 300\frac{s+\sqrt{d}}{s+10\sqrt{d}}. \quad (31)$$

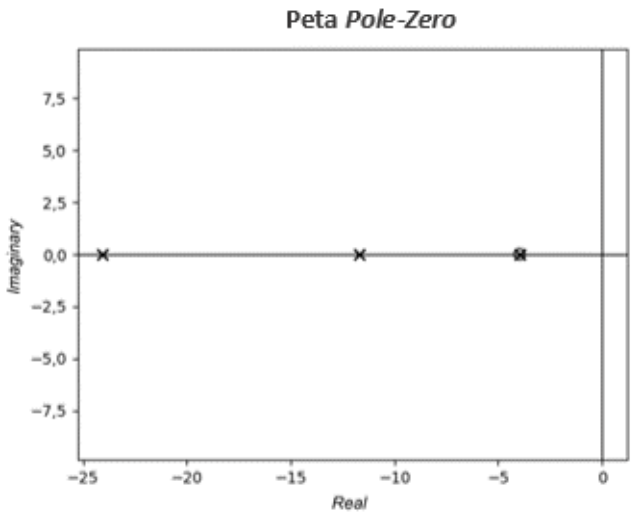
B. STATE COMPENSATOR

Dalam metode ini, dikembangkan *state compensator* dan *state estimator* berdasarkan model *inverted pendulum* yang dilinearisasi. Sinyal kendali umpan balik keadaan diberikan oleh (32).

$$u = -K\,\hat{x}_s \quad (32)$$

dengan K adalah konstanta umpan balik keadaan dan $\hat{x}$ adalah *state estimator* x. Penguatan umpan balik dipilih sedemikian rupa sehingga persamaan berikut ini berlaku.

$$det\left(sI - (A - BK)\right) = f(s) = 0 \quad (33)$$



Gambar 5. Peta *pole-zero* dari sistem kalang tertutup menggunakan pengendali berbasis *root locus*.

dengan $f(s)$ merupakan polinomial orde 4 yang mewakili persamaan karakteristik target dari sistem kalang tertutup yang *pole*-nya, $s_{c1}, s_{c2}, s_{c3}, s_{c4}$, dapat ditentukan sebagai berikut.

$$f(s) = (s - s_{c1})(s - s_{c2})(s - s_{c3})(s - s_{c4}) = 0.$$

Dipilih nilai berikut ini sebagai *pole* target.

$$s_{c1} = -1, s_{c2} = -2, s_{c3} = -3, s_{c4} = -4.$$

Nilai *state estimator* dapat diperbarui dengan menggunakan persamaan berikut.

$$\dot{\hat{x}}_s = A\hat{x}_s + Bu + L(y - C\hat{x}_s) \quad (34)$$

dengan L merupakan penguatan *estimator* yang dipilih sedemikian rupa sehingga persamaan berikut ini berlaku.

$$det\left(sI - (A - LC)\right) = g(s) = 0 \quad (35)$$

dengan $g(s)$ adalah polinomial orde 4 yang merepresentasikan persamaan karakteristik target dari dinamika *error estimator* yang *pole*-nya, $s_{e1}, s_{e2}, s_{e3}, s_{e4}$, dapat ditentukan sebagai berikut.

$$g(s) = (s - s_{e1})(s - s_{e2})(s - s_{e3})(s - s_{e4}) = 0. \quad (36)$$

Di sini, dipilih nilai berikut sebagai *pole* target, yang sepuluh kali lebih besar dari *pole* pengendali, untuk memastikan bahwa dinamika kesalahan *state estimator* lebih cepat konvergen.

$$s_{e1} = -10, s_{e2} = -20, s_{e3} = -30, s_{e4} = -40.$$

Diagram blok *state compensator* diperlihatkan pada Gambar 6.

C. KENDALI PD

Dalam metode ini, digunakan pengendali PD berikut ini untuk menghasilkan sinyal kendali.

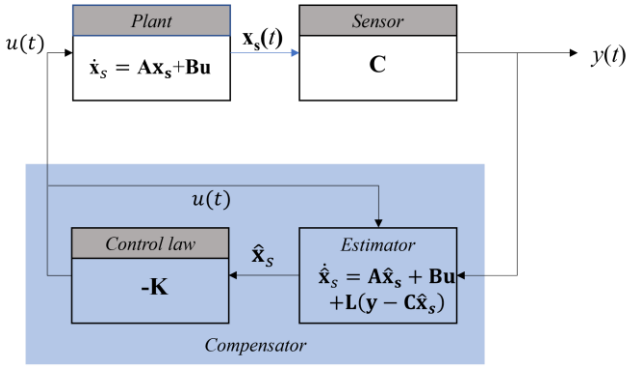
$$u = K_p\left(e + T_d\frac{de}{dt}\right). \quad (37)$$

Dengan transformasi sinyal kendali ke ruang frekuensi kompleks, diperoleh (38).

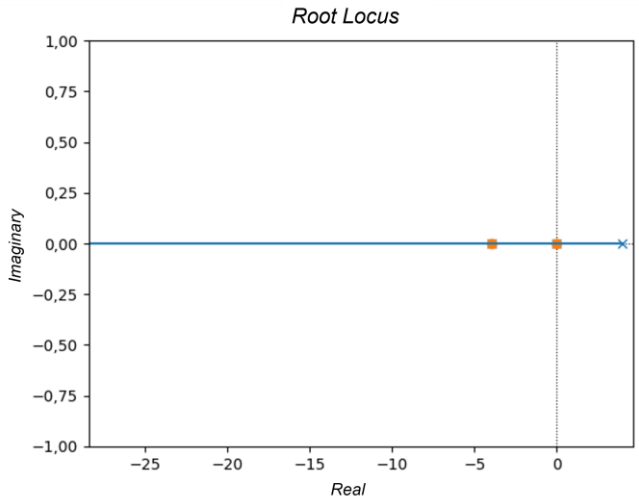
$$U(s) = K_p(1 + T_d s)E(s) \quad (38)$$

sehingga *zero* kendali PD terletak pada $s_{PD} = -\frac{1}{T_d}$.

Zero kendali PD dipilih untuk ditempatkan pada *pole* negatif *inverted pendulum*, $s_{PD} = s_2 = -\sqrt{d}$. Oleh karena itu,



Gambar 6. Diagram blok *state compensator*.



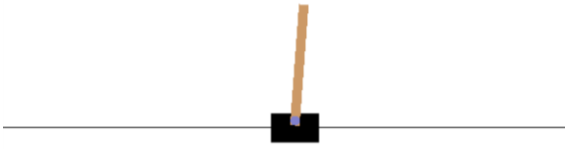
Gambar 7. *Root locus* dari sistem yang dimodifikasi setelah penambahan kendali PD.

root locus baru dari sistem akan bergerak ke arah setengah bidang kiri dari ruang frekuensi kompleks, alih-alih bergerak ke arah sumbu imajiner, seperti yang diilustrasikan pada Gambar 7. Kemudian, penguatan pengendali dapat dipilih untuk memastikan bahwa *pole* sistem kalang tertutup terletak di setengah bidang kiri. Untuk tujuan ini, penguatan $K_p = 30$ digunakan.

D. REINFORCEMENT LEARNING

Algoritma *reinforcement learning* yang digunakan di sini adalah PPO [15]. Algoritma ini termasuk dalam kategori *reinforcement learning* berbasis aturan yang berfokus pada pencarian aturan terbaik $\pi(a|s)$, yang merepresentasikan kemungkinan tindakan tertentu a untuk suatu keadaan tertentu s. Algoritma ini menggunakan pendekatan *actor-critic*, yaitu aktor memperkirakan aturan, sedangkan *critic* mengukur tingkat baiknya tindakan yang dilakukan.

Algoritma dimulai dengan dua jaringan, yang pertama untuk aturan dengan parameter θ dan yang kedua untuk fungsi nilai V dengan parameter ϕ. Algoritma dimulai dari iterasi $k = 0$ dengan mengumpulkan satu set lintasan $D_k = \{\tau\}$ dari $t = 0$ sampai $t = T$, dengan setiap lintasan terdiri atas $\tau = (s_0, a_0, s_1, a_1, \dots, s_T)$. Algoritma berjalan menggunakan aturan saat ini, $\pi_k = \pi(\theta_k)$. Dari lintasan tersebut, *reward* yang akan diberikan dihitung, $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$. Langkah selanjutnya adalah menghitung estimasi keuntungan $\hat{A}_t$, yang didefinisikan sebagai keuntungan memilih tindakan berdasarkan aturan saat ini π_k dibandingkan dengan pengambilan sampel tindakan secara acak. Fungsi keuntungan diestimasi berdasarkan fungsi nilai saat ini, $V_k = V(\phi_k)$.



Gambar 8. Tangkapan layar CartPole.

TABEL I
NILAI PARAMETER

Parameter	Nilai	Parameter	Nilai
m_c	1 kg	x_{th}	2,4 m
m_p	0,1 kg	θ_{th}	3π/45 rad
l	0,5 m	M	200 langkah

Kemudian, parameter jaringan aturan diperbarui menggunakan pendakian gradien stokastik dengan memaksimalkan fungsi objektif PPO sebagai berikut.

$$\theta_{k+1} = arg\max_{\theta} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}_t, g\left(\epsilon, \hat{A}_t\right) \right) \quad (39)$$

dengan fungsi $g(\epsilon, A)$ didefinisikan sebagai (40).

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A; & A \geq 0 \\ (1-\epsilon)A; & A < 0 \end{cases} \quad (40)$$

dengan ϵ merupakan *hyperparameter*.

Langkah selanjutnya adalah memperbarui parameter jaringan fungsi nilai dengan meminimalkan fungsi objektif *mean-squared error* (MSE) menggunakan penurunan gradien. MSE diberikan oleh (41).

$$\phi_{k+1} = arg\min_{\phi} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} (V_k(s_t) - \hat{R}_t)^2. \quad (41)$$

IV. SKENARIO PENGUJIAN

Untuk menguji kinerja algoritme kendali dalam model *inverted pendulum* pada gerobak, digunakan simulasi Gym Environment yang dimodifikasi, yang disebut CartPole, yang dikembangkan oleh OpenAI [16], seperti yang ditunjukkan pada Gambar 8. Lingkungan berbasis Python ini banyak digunakan dalam komunitas *reinforcement learning* karena sederhana dan juga karena lingkungan ini secara langsung menyediakan fungsi *reward* untuk pelatihan. Simulasi ini menggunakan pendekatan model nonlinear *inverted pendulum* seperti yang dijelaskan pada Bagian IIA dengan parameter yang ditunjukkan pada Tabel I. *Library* Python asli dimodifikasi untuk mendukung kendali atau aksi yang berkelanjutan. Keadaan asli pendulum dipilih secara acak menjadi $x_{c0} = 0{,}025$, $\dot{x}_{c0} = 0{,}009$, $\theta_0 = 0{,}043$, $\dot{\theta}_0 = 0{,}009$. Di antara keempat algoritma, PPO adalah satu-satunya algoritma yang bergantung pada pelatihan. Algoritma PPO dilatih selama 250.000 *epoch* dalam percobaan ini karena pada jumlah ini, *reward* rata-rata tampak konvergen ke nilai tinggi tertentu. Jadi, dapat diasumsikan bahwa algoritma *reinforcement learning* memiliki jumlah *epoch* yang cukup untuk mempelajari aturan yang paling optimal.

Kinerja algoritma kendali dibandingkan dalam hal respons transien dan respons kondisi tunak. Respons transien yang dipertimbangkan meliputi hal-hal sebagai berikut [17].

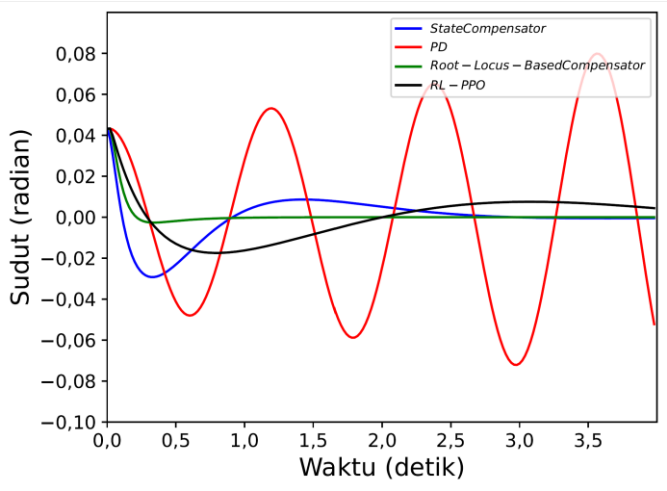
- *Overshoot*, yang didefinisikan sebagai

$$\%OV = \frac{\theta_{max} - \theta_{ss}}{\theta_{ss}} \times 100\%.$$

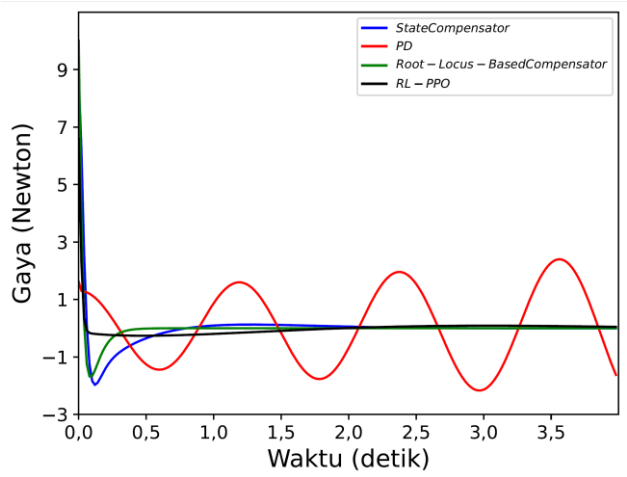
p-ISSN 2301–4156 | e-ISSN 2460–5719

TABEL II
PERBANDINGAN KINERJA

Parameter	*State Compensator*	PD	*Compensator* berbasis *Root Locus*	PPO
Overshoot (%)	70,74	122,15	6,11	61,17
Peak time (detik)	$3{,}20\times10^{-1}$	2,98	$3{,}40\times10^{-1}$	$8{,}00\times10^{-1}$
Settling time (detik)	-	-	$7{,}00\times10^{-1}$	-
Error kondisi tunak (rad)	$8{,}21\times10^{-4}$	$1{,}14\times10^{-2}$	$-5{,}48\times10^{-7}$	$5{,}60\times10^{-3}$
Jumlah energi (N^2)	$1{,}53\times10^{2}$	$3{,}49\times10^{2}$	$1{,}27\times10^{2}$	$5{,}24\times10^{1}$



Gambar 9. *Plot* sudut terhadap waktu yang dihasilkan oleh empat algoritma.



Gambar 10. *Plot* gaya terhadap waktu yang dihasilkan oleh empat algoritma.

Di sini, θ_{max} dan θ_{ss} menentukan nilai puncak keluaran dan nilai kondisi tunak keluaran.

- *Peak time*, yang didefinisikan sebagai jumlah waktu yang diperlukan untuk mencapai puncak pertama *overshoot*.
- *Settling time*, yang didefinisikan sebagai jumlah waktu yang diperlukan untuk mencapai dan tetap berada dalam kisaran 2% dari nilai kondisi tunak.

Di sisi lain, respons kondisi tunak meliputi hal-hal sebagai berikut.

- Kesalahan kondisi stabil, yang didefinisikan sebagai

$$e_{ss} = \theta_t - \theta_{ss}.$$

- Energi total selama langkah waktu tertentu, M, yang didefinisikan sebagai

$$W = \sum_{k=0}^{M} u_k^2.$$

V. HASIL DAN DISKUSI

A. KINERJA KENDALI

Plot sudut dan gaya sebagai fungsi waktu dapat diamati pada Gambar 9 dan Gambar 10. Garis biru, merah, hijau, dan hitam menunjukkan kinerja *state compensator*, pengendali PD, kompensator berbasis *root locus*, dan *reinforcement learning* berdasarkan algoritma PPO. Nilai-nilai parameter respons transien (*overshoot*, *peak time*, dan *settling time*) serta parameter respons kondisi tunak (*error* kondisi tunak dan jumlah energi) semuanya dirangkum dalam Tabel II.

Dari Gambar 9, dapat dilihat bahwa *state compensator*, *compensator* berbasis *root locus*, dan PPO mampu mengendalikan *inverted pendulum* menuju posisi vertikal ($\theta_t = 0$) dengan cara yang stabil. Akan tetapi, kendali PD tampaknya menghasilkan osilasi yang gagal mencapai nol. Pada Tabel II, dapat diamati juga bahwa semua algoritma kecuali *compensator* berbasis *root locus* gagal menetapkan nilai keluaran dalam kisaran 2% dari nilai kondisi tunak sebelum $M = 200$ langkah (yang sama dengan $T = 4$ detik). Hal ini ditunjukkan dengan tanda putus-putus pada bagian *settling time* untuk ketiga algoritma kecuali *compensator* berbasis *root locus*.

Dari Gambar 9 dan Tabel II, tampak bahwa di antara tiga metode yang berhasil, *compensator* berbasis *root locus* dapat dengan cepat memandu sudut pendulum menuju nol tanpa *overshoot* yang besar dan bekerja dengan *settling time* yang cepat. *State compensator*, meskipun memiliki *peak time* yang cepat, kinerjanya lebih buruk dalam hal *overshoot* dan *settling time* dibandingkan dengan compensator berbasis *root locus*. PPO memiliki kinerja lebih baik daripada *state compensatorr* dalam hal *overshoot*, tetapi lebih buruk dalam hal *settling time* dan *peak time* dibandingkan dengan dua metode lainnya. Alasan dari hal ini adalah fakta bahwa fungsi *reward* yang digunakan untuk memandu proses pembelajaran hanya berfokus pada aspek pendulum yang tidak jatuh. Perlu diingat bahwa *reward* bernilai +1 diberikan setiap kali sudut pendulum jatuh di dalam rentang tertentu. Hal ini berarti bahwa kinerja yang diperkuat tidak mencakup aspek *peak time* (waktu yang diperlukan sudut untuk bergerak menuju *setpoint* target), *settling time* (waktu yang diperlukan sudut tersebut untuk konvergen), atau *overshoot*.

Dari perspektif respons kondisi tunak, terutama dalam hal *steady-state error*, *compensator* berbasis *root locus* sekali lagi memiliki kinerja lebih baik daripada dua metode lain yang berhasil. Hal ini ditunjukkan dengan nilai kesalahan yang sangat kecil, seperti diperlihatkan pada Tabel II. PPO sekali lagi berada di urutan kedua, diikuti oleh *state compensator*. Namun, kinerja yang berbeda diamati dalam hal jumlah keseluruhan energi. Dari Gambar 10, dapat dilihat bahwa gaya yang dihasilkan oleh PPO dengan cepat mengecil tanpa banyak *overshoot* dibandingkan dengan gaya yang dihasilkan oleh *state compensator* dan *compensator* berbasis *root locus*.

p-ISSN 2301–4156 | e-ISSN 2460–5719

Keunggulan PPO ini disebabkan oleh fakta bahwa PPO hanya berfokus pada tidak jatuh. Target yang lebih longgar ini memungkinkan algoritma untuk menghasilkan lebih sedikit energi, seperti yang ditunjukkan pada Tabel II, dibandingkan dengan metode lainnya.

B. DISKUSI

Dari hasil sebelumnya, dapat dibandingkan cara metode *reinforcement learning*, dalam hal ini algoritma PPO, bekerja dibandingkan dengan metode kendali konvensional seperti *state compensator*, PD, dan *compensator* berbasis *root locus* dalam arti yang lebih luas. Dalam hal latar belakang pengetahuan yang diperlukan untuk menghasilkan aturan kendali, jelas bahwa metode *reinforcement learning* tidak memerlukan pengetahuan apa pun terkait model pendulum. Hal ini tidak terjadi pada metode lain, terutama *state compensator* dan *compensator* berbasis *root locus*. Keduanya sangat bergantung pada properti mekanis dan model dinamika *inverted pendulum*. Kendali PD, dalam beberapa hal, secara teoretis mampu bekerja dengan baik, meskipun model sistem tidak diketahui. Seperti yang telah dibahas secara luas dalam literatur [18], hal ini dapat terjadi melalui proses pengaturan parameter, seperti metode Ziegler-Nichols yang terkenal. Namun, tidak ada jaminan bahwa kinerja yang optimal (atau bahkan masuk akal) dapat dicapai dengan pendekatan semacam ini. Fakta bahwa metode ini selalu bergantung pada tiga parameter, untuk kasus proporsional-integral-derivatif (PID) juga membatasi kemungkinan aturan kendali. Di sisi lain, metode *reinforcement learning* bergantung pada fungsi aturan yang diwakili oleh jaringan saraf dalam kasus ini. Seperti yang telah dibahas dalam literatur, jaringan saraf tiruan adalah pendekatan fungsi yang dapat diandalkan yang bekerja untuk berbagai fungsi [19].

Namun, meskipun tidak memerlukan model sebelumnya, *reinforcement learning* bergantung pada percobaan atau perlu menerapkan aturan saat ini, $\pi(s)$, pada pendulum dan belajar dari fungsi *reward*. Pertama-tama, hal ini bisa menjadi proses yang rumit serta membutuhkan banyak waktu dan komputasi sebelum algoritma dapat menghasilkan aturan terbaik, $\pi^*(s)$, untuk menghasilkan kinerja yang masuk akal. Kedua, melakukan uji coba pada sebuah *plant* mungkin tidak praktis dalam beberapa kasus. Karena aturan yang diterapkan cenderung tidak optimal pada awalnya, aturan tersebut dapat menghasilkan pergerakan yang tidak terduga atau bahkan berbahaya.

Untuk menghindari masalah ini, beberapa penelitian telah dilakukan dengan berfokus pada pelaksanaan pelatihan dalam skenario simulasi sebelum implementasi [20]. Hal ini dapat dilakukan untuk beberapa kasus, termasuk *inverted pendulum* dalam kasus ini. Namun, simulator yang baik pada umumnya bergantung pada model berbasis fisika yang juga membutuhkan banyak komputasi.

Terlepas dari tantangan ini, muncul argumen bahwa metode kendali konvensional juga memiliki masalah yang sama. Hal ini terjadi karena metode seperti *state compensator* atau *compensator* berbasis *root locus* membutuhkan model sistem yang perlu diambil dari beberapa jenis eksperimen identifikasi sistem. Jenis eksperimen ini juga membutuhkan semacam "coba-coba".

Hal penting lainnya adalah fakta bahwa kinerja algoritma *reinforcement learning* bergantung pada fungsi *reward*. Tantangan dalam robotika, yaitu fungsi *reward* tidak tersedia secara umum, adalah menentukan fungsi *reward* yang paling mencerminkan kinerja yang diinginkan. Secara teori, ada banyak sekali kemungkinan di sini. Untuk kasus *inverted pendulum* ini, target membuat pendulum tetap berada di posisi vertikal dapat dicapai dengan berbagai fungsi *reward*. Hal ini dapat menjadi *reward* sederhana yang digunakan, dengan memberikan *reward* positif yang datar setiap kali sudutnya berada di bawah nilai tertentu. Namun, jenis *reward* ini tidak terlalu peduli dengan kinerja transien atau kinerja kondisi tunak seperti metode kendali konvensional lainnya. Jika benar-benar ingin dicapai kinerja berdasarkan beberapa pertimbangan tertentu, fungsi *reward* yang lebih kompleks diperlukan melalui proses yang disebut *reward shaping* [12], [21].

Jadi, dari poin-poin penting sebelumnya, dapat ditarik beberapa pertimbangan penting tentang waktu diperlukannya penggunaan *reinforcement learning* dalam aplikasi kendali umum. Ketika *plant* yang sedang dikendalikan hanya memberikan sedikit atau bahkan tidak ada informasi mengenai modelnya, tetapi pengujian aturan kendali cukup mudah, *reinforcement learning* dapat menjadi kandidat yang baik untuk diterapkan. Fakta bahwa metode ini tidak bergantung pada model sebelumnya dan penggunaan jaringan saraf sebagai pendekatan aturan kendali sangat kuat dibandingkan dengan metode kendali konvensional, bahkan kendali PID yang telah populer di industri, karena kemampuannya untuk bekerja tanpa model eksplisit dari *plant*. Jika pengujian aturan kendali mudah dan aman, akan lebih baik jika digunakan *reinforcement learning* karena memerlukan *trial and error* yang ekstensif pada *plant* yang dikendalikan. Penggunaan *reinforcement learning* lebih cocok lagi jika beberapa jenis fungsi tujuan tersedia, sehingga dapat dimodifikasi menjadi fungsi *reward* untuk memandu proses pembelajaran.

VI. KESIMPULAN

Dalam makalah ini, dilakukan studi perbandingan antara algoritma *reinforcement learning*, yaitu PPO, dan beberapa algoritma kendali konvensional, yaitu *compensator* berbasis *root locus*, *state compensator*, dan kendali PD, untuk mengendalikan *inverted pendulum*. Ditemukan bahwa dengan jumlah pelatihan yang cukup, algoritma PPO mampu mencapai kinerja transien dan kondisi tunak yang sebanding jika dibandingkan dengan metode kendali konvensional, meskipun informasi tentang model pendulum tidak diketahui. Kinerja algoritma ini dapat ditingkatkan lebih lanjut dengan melakukan teknik *reward shaping* untuk memasukkan spesifikasi kinerja ke dalam fungsi *reward*. Dari analisis hasil, dapat disimpulkan bahwa *reinforcement learning* paling cocok untuk masalah kendali, terutama jika tidak ada model sebelumnya dari *plant* yang tersedia dan melakukan *trial and error* terhadap model tersebut.

KONFLIK KEPENTINGAN

Para penulis menyatakan tidak ada konflik kepentingan dalam penelitian dan penulisan makalah ini.

KONTRIBUSI PENULIS

Konseptualisasi, Ahmad Ataka; metodologi, Ahmad Ataka, Dzuhri Radityo Utomo, dan Adha Imam Cahyadi; perangkat lunak, Ahmad Ataka; validasi, Ahmad Ataka dan Andreas Sandiwan; analisis formal, Ahmad Ataka dan Dzuhri Radityo Utomo; investigasi, Ahmad Ataka; sumber daya, Ahmad Ataka; kurasi data, Ahmad Ataka; penyusunan draf awal, Ahmad Ataka dan Andreas Sandiwan; tinjauan dan penyuntingan tulisan, Ahmad Ataka, Andreas Sandiwan, Hilton Tnunay, Dzuhri Radityo Utomo, dan Adha Imam

Cahyadi; visualisasi, Ahmad Ataka dan Andreas Sandiwan; pengawasan, Hilton Tnunay, Dzuhri Radityo Utomo, dan Adha Imam Cahyadi; administrasi proyek, Ahmad Ataka dan Andreas Sandiwan; perolehan dana, Ahmad Ataka, Dzuhri Radityo Utomo, dan Adha Imam Cahyadi.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh mahasiswa S1 yang mengambil mata kuliah Teknik Kendali, Teknik Kendali Modern, dan Robotika Bergerak pada Semester Ganjil 2022 atas karya-karya briliannya dalam pengendalian *inverted pendulum* yang menjadi motivasi dalam penulisan makalah ini.

REFERENCES

[1] R.S. Sutton dan A.G. Barto, *Reinforcement Learning: An Introduction*. London, Inggris: The MIT Press, 2020.

[2] A. Krizhevsky, I. Sutskever, dan G.E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, Vol. 60, No. 6, hal. 84–90, Jun. 2017, doi: 10.1145/3065386.

[3] V. Mnih dkk., "Playing Atari with Deep Reinforcement Learning," 2013, *arXiv:1312.5602*.

[4] V. Mnih dkk., "Human-Level Control Through Deep Reinforcement Learning," *Nat.*, Vol. 518, hal. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[5] D. Silver dkk., "A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go Through Self-Play," *Sci.*, Vol. 362, No. 6419, hal. 1140–1144, Des. 2018, doi: 10.1126/science.aar6404.

[6] Ardiansyah dan E. Rainarli, "Implementasi Q-Learning dan Backpropagation pada Agen yang Memainkan Permainan Flappy Bird," *J. Nas. Tek. Elekt., Teknol. Inf.*, Vol. 6, No. 1, hal. 1–7, Feb. 2017, doi: 10.22146/jnteti.v6i1.287.

[7] C. Berner dkk., "Dota 2 with Large Scale Deep Reinforcement Learning," 2019, *arXiv:1912.06680*.

[8] B.R. Kiran dkk., "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Trans. Intell. Transp. Syst.*, Vol. 23, No. 6, hal. 4909–4926, Jun. 2022, doi: 10.1109/TITS.2021.3054625.

[9] X. Ruan, D. Ren, X. Zhu, dan J. Huang, "Mobile Robot Navigation Based on Deep Reinforcement Learning," *2019 Chin. Control, Decis. Conf. (CCDC)*, 2019, hal. 6174–6178, doi: 10.1109/CCDC.2019.8832393.

[10] V. Tsounis dkk., "DeepGait: Planning and Control of Quadrupedal Gaits Using Deep Reinforcement Learning," *IEEE Robot., Automat. Lett.*, Vol. 5, No. 2, hal. 3699–3706, Apr. 2020, doi: 10.1109/LRA.2020.2979660.

[11] R. Liu dkk., "Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focussed Mini-Review," *Robot.*, Vol. 10, No. 1, hal. 1–13, Jan. 2021, doi: doi.org/10.3390/robotics10010022.

[12] J. Ibarz dkk., "How to Train Your Robot with Deep Reinforcement Learning: Lessons We Have Learned," *Int. J. Robot. Res.*, Vol. 40, No. 4–5, hal. 698–721, Jan. 2021, doi:10.1177/0278364920987859.

[13] A. Sharma, R. Ahmad, dan C. Finn, "A State-Distribution Matching Approach to Non-Episodic Reinforcement Learning," 2022, *arXiv:2205.05212*.

[14] A.G. Barto, R.S. Sutton, dan C.W. Anderson, "Neuronlike Adaptive Elements that Can Solve Difficult Learning Control Problems," *IEEE Trans. Syst. Man, Cybern.*, Vol. SMC-13, No. 5, hal. 834–846, Sept.-Okt. 1983, doi: 10.1109/TSMC.1983.6313077.

[15] J. Schulman *dkk.*, "Proximal Policy Optimization Algorithms," 2017, *arXiv:1707.06347*.

[16] G. Brockman *dkk.*, "OpenAI Gym," 2016, *arXiv:1606.01540*.

[17] R.P. Borase, D.K. Maghade, S.Y. Sondkar, dan S.N. Pawar, "A Review of PID Control, Tuning Methods and Applications," *Int. J. Dyn., Control*, Vol. 9, No. 2, hal. 818–827, Jun. 2021, doi: 10.1007/s40435-020-00665-4.

[18] N.S. Nise, *Control Systems Engineering*, 4th ed. Hoboken, AS: John Wiley & Sons, Inc., 2004.

[19] S. Liang dan R. Srikant, "Why Deep Neural Networks for Function Approximation?" 2017, *arXiv:1610.04161*.

[20] W. Zhao, J.P. Queralta, dan T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," *2020 IEEE Symp. Ser. Comput. Intell. (SSCI)*, 2020, hal. 737–744.

[21] J. Eschmann, "Reward Function Design in Reinforcement Learning," dalam *Reinforcement Learning Algorithms: Analysis and Applications*, B. Belousov dkk., Eds., Cham, Swiss: Springer Cham, 2021, hal. 25–33.