Jurnal Algoritma, Logika dan Komputasi *Vol.IX (No. 02) : 25 - 37. Th. 2026*
Versi Online: https://journal.ubm.ac.id/index.php/alu *p-ISSN: 2620-620X*
DOI: http://dx.doi.org/10.30813/j-alu.v2i2.10104 *e-ISSN: 2621-9840*

IMPLEMENTASI ALGORITMA RECIPROCAL VELOCITY OBSTACLES (RVO) UNTUK PERGERAKAN KERUMUNAN MUSUH GAME ROGUELIKE SURVIVAL

IMPLEMENTATION OF RECIPROCAL VELOCITY OBSTACLES (RVO) ALGORITHM FOR ENEMY CROWD MOVEMENT IN ROGUELIKE SURVIVAL GAME

Jonathan Salim, jonathansalim6@gmail.com [1)], Dionisia Bhisetya Rarasati, dionisia.rarasati@gmail.com [2)]
[1) 2)] Fakultas Teknologi dan Desain, Universitas Bunda Mulia

Diterima 10 Juni 2026 / Disetujui 24 Agustus 2026

ABSTRACT

Roguelike survival video games require a navigation system capable of handling hundreds of enemy entities simultaneously without compromising visual quality or performance stability. The primary technical challenges include agent overlapping and movement stuttering (jittering). This study aims to implement the Reciprocal Velocity Obstacles (RVO) algorithm to manage enemy crowd movement, ensuring it appears natural and responsive. The research methodology involves designing a navigation system using the RVO algorithm optimized with Spatial Grid-based Neighbor Selection techniques to limit the calculation range of agents. The evaluation was conducted in two main phases: performance analysis (Frame Per Second and performance retention) across 10 enemy waves using Benchmarking method and User Acceptance Test (UAT) evaluation. Technical results using Benchmarking method demonstrated high system stability, with a performance retention value of 82.93% at peak load in wave 10, involving 115 enemy agents, which remained above the 70% minimum feasibility threshold. The UAT evaluation results recorded a 92.85% success rate in preventing overlapping and 90.47% in minimizing jittering. The assessment of movement naturalness reached 88.57%, categorizing the system as "Very Good." In conclusion, the RVO algorithm is a solution for crowd management on minimum specification devices, as it produces smooth, realistic, and efficient movement.

Keywords: *Reciprocal Velocity Obstacles, Benchmarking, Roguelike Survival, Game Performance, User Acceptance Test.*

ABSTRAK

Permainan video bergenre *roguelike survival* menuntut sistem navigasi yang mampu menangani ratusan entitas musuh secara bersamaan tanpa mengorbankan kualitas visual maupun stabilitas performa. Masalah utama yang sering muncul adalah terjadinya tumpang tindih (*overlapping*) antar agen dan getaran pergerakan (*jittering*). Penelitian ini bertujuan untuk mengimplementasikan algoritma *Reciprocal Velocity Obstacles* (RVO) untuk mengatur pergerakan kerumunan musuh terlihat alami dan responsif. Metodologi penelitian mencakup perancangan sistem navigasi menggunakan algoritma RVO yang dioptimasi dengan teknik *Neighbor Selection* berbasis *Spatial Grid* untuk membatasi jangkauan kalkulasi agen. Pengujian dilakukan melalui dua tahap utama, yaitu analisis performa teknis (*Frame Per Second* dan retensi performa) pada 10 gelombang musuh menggunakan metode *Benchmarking* serta evaluasi *User Acceptance Test* (UAT). Hasil pengujian *Benchmarking* menunjukkan stabilitas sistem yang tinggi, dengan nilai retensi performa sebesar 82,93% pada kondisi beban puncak *wave* 10 yang melibatkan 115 agen musuh, yang berada di atas ambang batas kelayakan minimum 70%. Hasil evaluasi UAT mencatat tingkat keberhasilan sebesar 92,85% dalam mencegah *overlapping* dan 90,47% dalam meminimalkan *jittering*. Penilaian terhadap aspek kealamian pergerakan mencapai 88,57%, yang mengkategorikan sistem dalam rentang Sangat Baik. Simpulan penelitian ini menunjukan bahwa algoritma RVO merupakan solusi untuk manajemen kerumunan pada perangkat berspesifikasi minimum karena mampu menghasilkan pergerakan yang mulus, realistis, dan efisien.

Kata Kunci: *Reciprocal Velocity Obstacles*, *Benchmarking*, *Roguelike Survival*, Performa *Game*, *User Acceptance Test*.

*Korespondensi Penulis:
E-mail: jonathansalim6@gmail.com

PENDAHULUAN

Game merupakan salah satu media elektronik yang bertujuan untuk menghibur, sehingga pengembang game dituntut untuk membuat *game* yang menarik [1]. Kini industri pengembangan *game* menuntut tingkat interaktivitas tinggi, terutama pada perilaku *Non Player Character* (NPC). NPC berperan sebagai elemen kunci yang memberikan tantangan dinamis dan menghidupkan suasana permainan melalui interaksi dengan pemain [2]. Salah satu komponen paling krusial dalam implementasi kecerdasan buatan (*Artificial Intelligence*) dalam *game* adalah navigasi dan pencarian jalur (*pathfinding*). NPC harus bergerak secara adaptif terhadap kondisi lingkungan, sehingga kegagalan merancang sistem navigasi menyebabkan pergerakan NPC menjadi kaku dan repetitif [3]. Perancangan sistem navigasi NPC yang cerdas dan efisien menjadi kebutuhan mendesak, terutama pada platform perangkat dengan sumber daya terbatas.

Dalam *game* bergenre *roguelike survival* yang seringkali menggunakan mekanisme musuh yang bergerombol (*horde*), pemain dihadapkan pada puluhan hingga ratusan musuh yang muncul secara bersamaan di dalam *game*. Metode pencarian jalur konvensional seperti A* (A-Star) efektif mencari rute statis, tetapi mengalami penurunan performa signifikan dan menghasilkan visual tidak alami pada kerumunan padat [4]. Penerapan metode penghindaran tabrakan berbasis gaya (*force-based*) seperti algoritma *Flocking* sering kali memicu fenomena osilasi atau getaran (*jittering*) pada agen ketika tingkat kepadatan kerumunan meningkat [5]. Hal ini terjadi karena adanya konflik gaya tolak-menolak yang merusak estetika visual dan membebani komputasi *Central Processing Unit* (CPU) akibat kalkulasi gaya fisik yang terus-menerus [6].

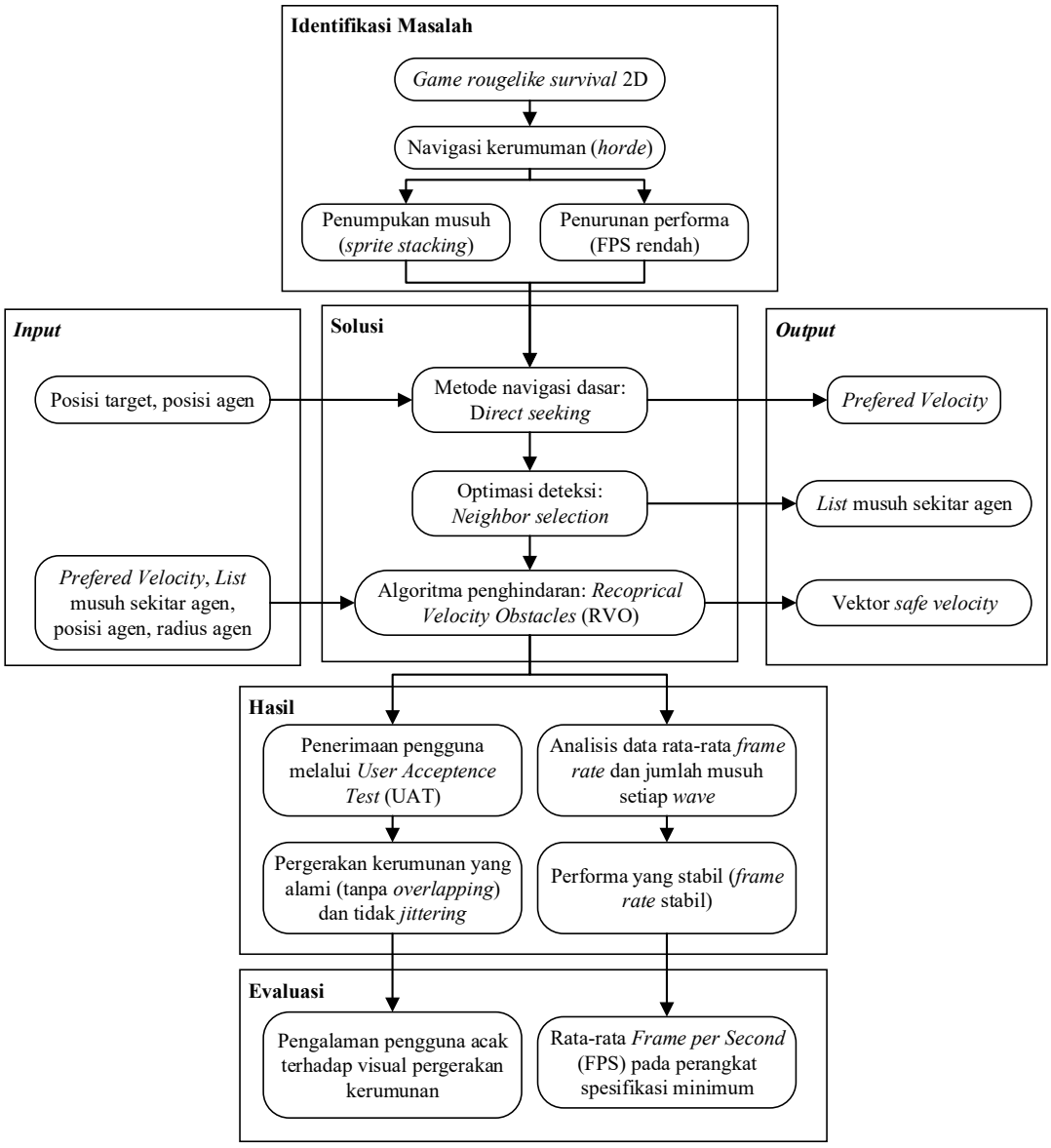
Algoritma *Reciprocal Velocity Obstacles* (RVO) menawarkan pendekatan navigasi yang terbukti lebih efisien dalam menangani kepadatan kerumunan dinamis. Implementasi simulasi kerumunan menunjukkan bahwa penerapan RVO mampu menjaga stabilitas *frame rate* meskipun jumlah agen meningkat, serta menghilangkan fenomena osilasi gerak melalui mekanisme prediksi kecepatan timbal balik [7]. Meskipun studi implementasi pada simulasi kerumunan menggunakan metode *Flocking* mampu menciptakan perilaku kelompok yang interaktif pada skenario standar, algoritma RVO menjadi krusial dalam skenario berdensitas tinggi untuk menghilangkan getaran (*jittering*) dan menjamin efisiensi komputasi yang lebih stabil dibandingkan metode navigasi tradisional [8].

Penelitian ini mengusulkan pendekatan navigasi hibrida (*hybrid*) yang menggabungkan metode *Direct Seeking* untuk penentuan arah global dan algoritma RVO untuk penghindaran tabrakan lokal. Identifikasi masalah utama berfokus pada tingkat penerimaan pengguna terhadap implementasi algoritma RVO dalam menghasilkan pergerakan kerumunan yang alami menggunakan *User Acceptance Test* (UAT). Penelitian ini mengimplementasikan algoritma RVO dengan zona tabrakan berbentuk lingkaran pada *game engine* Unity. Sistem ini mengeliminasi operasi trigonometri dan operasi akar yang pada kalkulasi zona tabrakan berbentuk kerucut pada RVO standar untuk memastikan karakter musuh dapat bernavigasi secara stabil tanpa menurunkan performa pada perangkat berspesifikasi minimum. Penelitian ini juga mengintegrasikan teknik optimasi *Neighbor Selection* berbasis *Spatial Grid* untuk mengatasi masalah *overhead* komputasi pada *game engine* saat menangani banyak unit. Oleh karena itu, pengembangan *game* di penelitian ini dilakukan dengan mengintegrasikan ketiga komponen tersebut bertujuan untuk menciptakan sistem navigasi yang menjamin stabilitas kinerja (*frame rate*) yang efisien pada *game* bergenre *roguelike survival* 2D.

METODE PENELITIAN

Rancangan penelitian berfokus pada integrasi arsitektur navigasi hibrida (*hybrid*) untuk mengendalikan pergerakan kerumunan agen musuh. Sitem menggunakan metode *Direct Seeking* untuk menentukan arah pergerakan global menuju posisi koordinat pemain. Kemudian sistem menerapkan algoritma RVO sebagai pengendali navigasi lokal untuk menghitung ruang kecepatan aman. Algoritma RVO memotong ruang kecepatan potensial yang dapat memicu

tabrakan sebelum terjadi kontak fisik antar agen musuh. Pendekatan prediktif tersebut mengasumsikan bahwa setiap agen melakukan upaya penghindaran yang setara. Alur tahapan penelitian disajikan dalam bentuk gambar yang dapat dilihat pada Gambar 1.



Gambar 1. Alur Tahapan Penelitian

Direct Seeking

Metode *Direct Seeking* adalah teknik dasar dalam navigasi otonom yang memungkinkan agen untuk bergerak lurus menuju posisi target yang telah ditentukan. Dalam pengembangan sistem *Non-Playable Character* (NPC) yang responsif, metode *Direct Seeking* berfungsi sebagai penentu kecepatan awal (*preferred velocity*) sebelum dimodifikasi oleh algoritma RVO. Perhitungan metode *Direct Seeking* berbasis pada operasi vektor sederhana yang didefinisikan sebagai Persamaan 1 dan 2:

$$\boldsymbol{D} = P_{target} - P_{agen} \quad \text{.....(1)}$$

$$\boldsymbol{V_{pref}} = \frac{D}{|D|} \times S_{maks} \quad \text{.....(2)}$$

Meskipun metode ini menjamin rute terpendek secara geometris, *Direct Seeking* memiliki kelemahan jika hanya digunakan sendirian pada lingkungan padat. Metode *Direct Seeking* tidak memiliki mekanisme deteksi tabrakan, sehingga agen akan terus bergerak maju meskipun ada tembok atau agen lain di depannya yang menyebabkan tumpang tindih (*overlapping*) visual. Dalam pangembangan *game*, metode *Direct Seeking* biasanya dikombinasikan dengan algoritma navigasi adaptif (seperti A* atau RVO) agar NPC mampu mengejar target tanpa mengabaikan rintangan di sekitarnya [9].

Neighbor Selection

Masalah utama pada simulasi kerumunan berskala besar adalah beban komputasi yang meningkat secara eksponensial seiring bertambahnya jumlah objek [6]. Dalam simulasi kerumunan tanpa partisi ruang, sistem harus membandingkan jarak setiap agen dengan seluruh agen lainnya (*all-pair checks*), yang menghasilkan kompleksitas waktu $O(N^2)$. Untuk mengatasi permasalahan skalabilitas tersebut, penerapan algoritma RVO memerlukan batasan persepsi lokal, di mana setiap agen hanya akan memproses agen lain yang berada di area sekitarnya [10]. Prosedur penelitian ini menerapkan teknik seleksi tetangga berbasis *Spatial Grid* untuk mengatasi kendala tersebut. Arena permainan dibagi menjadi struktur petak *grid* dua dimensi dimana ukuran setiap sel *grid* disesuaikan dengan radius deteksi maksimum dari agen musuh. Mekanisme ini membatasi jangkauan pencarian tetangga terdekat, dimana agen hanya memproses kalkulasi interaksi RVO dengan agen lain yang berada di dalam sel *grid* yang sama atau sel yang bersebelahan. Rumus pemetaan posisi agen $P(x, y)$ ke indeks *grid* (i, j) didefinisikan sebagai Persamaan 3:

$$i = \left\lfloor \frac{P_x}{C} \right\rfloor, \quad j = \left\lfloor \frac{P_y}{C} \right\rfloor \text{.....(3)}$$

Melalui perhitungan indeks dengan Persamaan 3, mekanisme pencarian tetangga (*neighbor selection*) dapat dilakukan secara lokal tanpa perlu memindai seluruh populasi agen secara global. Ruang lingkup pemeriksaan dibatasi hanya pada agen-agen yang berada dalam sel tempat agen berada serta sel-sel tetangga yang berbatasan langsung. Dalam konteks ruang dua dimensi, area pencarian ini tereduksi menjadi maksimal sembilan sel *grid* (*Moore Neighborhood*). Pembatasan area pencarian mampu mengurangi beban komputasi deteksi tabrakan secara signifikan dan menjamin stabilitas performa sistem dibandingkan metode pemeriksaan konvensional.

Reciprocal Velocity Obstacles (RVO)

Formulasi matematika algoritma RVO bekerja pada ruang kecepatan untuk memprediksi dan menghindari tabrakan sebelum kontak fisik terjadi. Algoritma RVO dibangun berdasarkan konsep *Velocity Obstacle* (VO) yang memetakan seluruh kecepatan agen yang dapat memicu tabrakan di masa depan [6]. Geometri VO menentukan himpunan semua kecepatan agen A yang akan memicu tabrakan dengan agen B pada masa depan. Persamaan dasar VO didefinisikan sebagai Persamaan 4:

$$\boldsymbol{VO_{A|B(v_B)}} = \{v_A \mid \lambda(p_A, v_A - v_B) \cap (B \oplus A) \neq \emptyset\} \text{.....(4)}$$

Algoritma RVO menyempurnakan kelemahan VO yang sering memicu *jittering* [10]. RVO membagi tanggung jawab penghindaran secara merata sebesar lima puluh persen (50%) kepada kedua agen yang saling berhadapan [7]. Agen A mengasumsikan bahwa agen B juga akan mengubah kecepatannya demi menghindari tabrakan. Rumus pemilihan kecepatan baru Agen A pada algoritma RVO dirumuskan sebagai Persamaan 5:

$$\boldsymbol{RVO_{A|B(v_B, v_A)}} = \{v'_A \mid v'_A = v_A + \frac{1}{2}(v^*_A - v_A), v^*_A \in VO_{A|B(v_B)}\} \text{.....(5)}$$

Variabel v_A^* menunjukkan kecepatan aman di luar ruang VO yang paling dekat dengan kecepatan optimal agen A. Melalui pembagian nilai setengah tersebut, agen A tidak mengeksekusi manuver ekstrem secara sepihak. Sistem mengambil nilai tengah antara kecepatan saat ini dan kecepatan penghindaran yang dibutuhkan.

Algoritma RVO dapat disederhanakan dari area kerucut menjadi sebuah area lingkaran pada titik prediksi tabrakan. Penyederhanaan ini mengurangi beban pemrosesan operasi trigonometri kompleks pada CPU, terutama dalam sistem *game* yang juga harus mengeksekusi berbagai sistem dan perhitungan. Algoritma RVO mendeteksi tabrakan dengan mengecek kuadrat jarak vektor kecepatan relatif terhadap radius lingkaran. Sistem menghitung vektor penghindaran dengan memproyeksikan kecepatan menjauhi titik pusat lingkaran. Pendekatan ini lebih efisien dalam komputasi tingkat tinggi jika diimplementasikan dalam *game* dengan jumlah agen yang banyak.

Pengujian

Tahap pengujian merupakan cara untuk memastikan bahwa sistem RVO dan optimasi *Spatial Grid* yang diimplementasikan dapat berjalan sesuai dengan kebutuhaan. Prosedur pengujian melibatkan partisipan (*tester*) yang menggunakan perangkat komputer yang memenuhi perangkat spesifikasi minimum untuk mensimulasikan kondisi penggunaan dunia nyata. Pengujian dirancang menggunakan pendekatan campuran (*mixed-method*), yaitu penggabungan antara pengambilan data teknis secara otomatis (*automated data logging*) untuk melakukan *benchmarking* dan survei pengguna melalui *User Acceptence Test* (UAT) untuk menilai penerimaan pengguna terhadap *game* yang sudah dibuat. Setiap tester diinstruksikan untuk memainkan *game* hingga tester menyelesaikan 10 *wave* sebelum mengisi formulir pengujian. Spesifikasi minimum yang diperlukan *tester* untuk memainkan *game* adalah:

Processor : Intel Core i3 4th Gen / AMD A8 Series atau setara
RAM : 4 GB
Graphics : Intel HD Graphics 520 / AMD Radeon Vega 3 atau kartu grafis setara yang mendukung DirectX 11.
Storage : 150 MB ruang kosong

1) *Benchmarking*

Simulasi kerumunan dan navigasi agen membutuhkan pengujian performa perangkat keras untuk memastikan sistem simulasi atau navigasi yang dibangun tidak menurunkan kinerja komputasi secara signifikan. Parameter utama pengujian ini adalah *Frame Per Second* (FPS). FPS didefinisikan sebagai satuan yang menunjukkan jumlah *frame* yang di-*render* oleh sistem dalam satu detik [11]. Batas kelayakan minimum *game* agar dapat dimainkan dengan nyaman adalah 60 FPS [12]. Data pengujian dapat dinormalisasi menjadi persentase retensi performa untuk menghilangkan bias dari perbedaan spesifikasi perangkat keras pengguna. Perhitungan retensi performa menggunakan Persamaan 6:

$$\boldsymbol{Retensi\ Performa} = \left(\frac{FPS_{sekarang}}{FPS_{awal}}\right) \times 100\% \quad \text{.....(6)}$$

Sistem diklasifikasikan memiliki stabilitas yang baik apabila persentase retensi performa bertahan di atas 70% pada kondisi *game* paling intens [13]. Artinya, penurunan performa atau toleransi *drop* FPS maksimal sebesar 30% dari FPS awal masih dianggap sebagai batas wajar.

2) Evaluasi *User Acceptance Test* (UAT)

Pengujian tahap akhir menggunakan *User Acceptance Test* (UAT) yang melibatkan pengguna akhir (*end user*) untuk memastikan sistem memenuhi ekspektasi pemain [14]. Instrumen kuesioner menggunakan metode Skala Likert dengan dua format rentang nilai. Evaluasi mengenai tumpang tindih dan getaran visual menggunakan skala genap satu hingga empat. Pemilihan skala genap dilakukan untuk menghilangkan opsi netral guna mencegah

kecenderungan responden memilih jawaban aman (*central tendency bias*). Empat metrik intensitas kejadian digunakan pada penilaian skala empat poin:

- Tidak Pernah: Masalah visual sama sekali tidak terlihat di layar sepanjang *game*. Pilihan ini merupakan kondisi paling ideal dan diberikan bobot empat poin.
- Jarang: Masalah visual terlihat satu atau dua kali namun tidak merusak fokus visual *game*. Pilihan ini diberikan bobot tiga poin.
- Sering: Masalah visual cukup banyak terjadi dan mulai mengganggu visual *game*. Pilihan ini diberikan bobot dua poin.
- Selalu: Masalah visual terus menerus terjadi dan sangat merusak pengalaman bermain. Pilihan ini merupakan indikator kegagalan sistem dan diberikan bobot satu poin.

Rentang angka satu hingga lima digunakan pada penilaian skala lima poin tanpa deskripsi teks spesifik pada setiap angkanya. Angka satu merepresentasikan pergerakan kerumunan yang sangat kaku dan tidak alami dan diberikan bobot terendah yaitu satu poin. Angka lima merepresentasikan pergerakan kerumunan yang sangat mulus dan terlihat alami dan merupakan kondisi paling ideal, sehingga diberikan bobot tertinggi yaitu lima poin. Angka dua, tiga, dan empat menunjukkan gradasi tingkat kealamian transisi pergerakan di antara kedua kondisi ekstrem tersebut. Nilai yang semakin tinggi menunjukkan kualitas pergerakan yang semakin baik. Evaluasi mengenai kealamian pergerakan menggunakan skala ganjil satu hingga lima. Tabel 1 menyajikan daftar pertanyaan yang wajib dijawab oleh *end user* setelah bermain *game*:

Tabel 1. Daftar Pertanyaan UAT

No	**Kategori**	
1	No	Kualitas Visual
	1	Apakah karakter musuh terlihat saling menembus fisik/badan satu sama lain (*clipping/overlapping*) saat mereka mengejar Anda dalam kelompok besar?
	2	Apakah pergerakan musuh terlihat bergetar (*jittering*) saat mereka berusaha mencari jalan mendekati Anda?
2	No	Kealamian Pergerakan
	1	Berikan penilaian skala 1 hingga 5 mengenai tingkat kealamian pergerakan kerumunan musuh dalam *game* ini.

Hasil akhir dari pengujian UAT berupa persentase penerimaan pengguna. Hasil persentase akhir didapatkan menggunakan Persamaan 7:

$$\boldsymbol{Persentase} = \left(\frac{Skor\ Aktual}{Skor\ Ideal}\right) \times 100\% \quad (7)$$

Pengujian UAT pada penelitian ini bertujuan untuk mengukur penilaian subjektif *end user* terhadap implementasi algoritma RVO, terutama pada aspek visual dan kealamian pergerakan agen musuh selama *game* berlangsung. Persentase akhir tersebut kemudian dipetakan ke dalam kategori tingkat keberhasilan. Tabel 2 menampilkan klasifikasi rentang persentase untuk evaluasi sistem.

Tabel 2. Kriteria Persentase Penerimaan

Rentang Persentase	**Interpretasi**
0% - 20%	Sangat Kurang
21% - 40%	Kurang
41% - 60%	Cukup
61% - 80%	Baik
81% - 100%	Sangat Baik

Kategori penilaian pada Tabel 2 menjadi acuan utama penelitian yang berfungsi untuk menarik kesimpulan akhir dari pengujian penerimaan pengguna. Implementasi algoritma RVO dinyatakan berhasil apabila hasil persentase mencapai rentang Baik atau Sangat Baik. Pencapaian pada rentang tersebut membuktikan bahwa penelitian telah memenuhi standar penerimaan

Versi Online: https://journal.ubm.ac.id/index.php/alu *p-ISSN: 2620-620X*
DOI: http://dx.doi.org/10.30813/j-alu.v2i2.10104 *e-ISSN: 2621-9840*

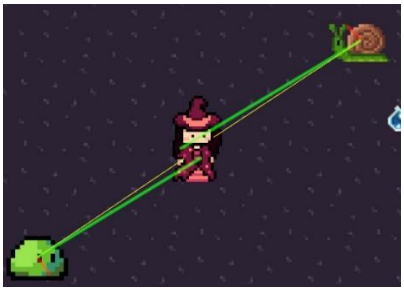
pemain. Penelitian yang dilakukan wajib mengevaluasi ulang parameter algoritma apabila hasil pengujian berada pada rentang Cukup atau lebih rendah.

HASIL DAN PEMBAHASAN

Implementasi sistem dilakukan menggunakan Unity sebagai *game engine* yang digunakan untuk pengembangan *game*. Lingkungan *game* dirancang sebagai area terbuka (*open map*) tanpa rintangan statis, sehingga kalkulasi algoritma difokuskan pada penghindaran tabrakan antar sesama agen musuh. Tahap implementasi bertujuan untuk memastikan algoritma RVO berjalan dengan baik dan sesuai dengan spesifikasi rancangan. Penelitian ini melakukan validasi visual untuk membuktikan bahwa seluruh agen musuh dapat bergerak dan menghindar secara responsif. Proses pengecekan implementasi ini menggunakan fitur *Gizmos* pada Unity Editor sebagai garis bantu visual. Komponen *Gizmos* memproyeksikan vektor ruang kecepatan dan batas fisik antar agen secara *real-time* selama simulasi berjalan. Indikator visual tersebut menampilkan garis bantu visual dari kalkulasi internal sistem. Terdapat tiga indikator warna utama pada garis *Gizmos* yang dievaluasi:

1) Garis kuning*:* Vektor kecepatan asli (*Direct Seeking*) menuju target.
2) Garis merah: Vektor penalti (*avoidance vector*) yang dihasilkan untuk menjauhi ancaman tabrakan.
3) Garis hijau*:* Vektor kecepatan akhir (*final velocity*) setelah algoritma menggabungkan arah pencarian utama dengan vektor penalti.

Algoritma RVO diimplementasikan ke dalam *game* menggunakan *game engine* Unity dengan beberapa skenario uji:

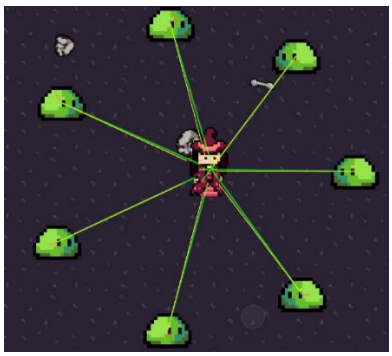


Gambar 2. Skenario Berhadapan

Gambar 3. Skenario Sudut 90 Derajat

Skenario berhadapan pada Gambar 2 menempatkan dua agen musuh yang bergerak saling berhadapan pada satu garis lurus. Hasil dari skenario berhadapan membuktikan bahwa prinsip

resiprositas (*reciprocity*) bekerja dengan baik. Sistem berhasil memberikan setengah beban koreksi penalti kepada agen pertama dan setengah beban kepada agen kedua dengan arah berlawanan. Kedua agen berhasil menghindari tabrakan secara adil tanpa ada agen yang mengorbankan jalurnya.



Gambar 4. Skenario Pengepungan

Gambar 5. Skenario Pengepungan

Gambar 3 menunjukan skenario dua agen saling berhadapan dengan sudut 90 derajat. Hasil dari skenario ini membuktikan bahwa algoritma RVO berhasil menghitung arah penghindaran agar agen tidak bertabrakan. Gambar 3 menunjukan garis hijau (vektor *final velocity*) pada kedua agen memanjang dan membelok dari rute utamanya, mengindikasikan kedua agen tidak ada yang mengurangi kecepatannya. Perilaku ini merupakan hasil dari algoritma RVO berbasis impuls. Algoritma RVO menambahkan vektor penalti secara langsung ke vektor kecepatan asli untuk mendorong agen menjauhi pusat area tabrakan. Hal ini mengakibatkan panjang garis kecepatan akhir agen meningkat untuk mempercepat diri keluar dari zona bahaya.

Gambar 4 melibatkan sekumpulan agen musuh yang mengepung posisi pemain dari segala arah. Skenario pengepungan menunjukkan perpotongan matriks garis hijau (*final velocity*) yang saling mengunci dan membentuk formasi melingkar di luar batas radius pemain. Algoritma RVO menahan langkah agen agar berhenti tepat di batas jangkauan tanpa mendesak dan menabrak agen lain di sebelahnya. Hasil ini membuktikan bahwa sistem RVO mampu menangani kalkulasi secara kumulatif.

Gambar 5 menggambarkan skenario kerumunan dimana agen yang berkumpul dan bergerak dinamis mengejar pemain di satu sisi area permainan. Skenario ini menunjukan posisi letak gambar (*sprite*) setiap agen secara fisik tetap terdistribusi dengan baik dan memiliki ruang sendiri. Algoritma terbukti sanggup menahan desakan ruang berlebih dan menjaga integritas bentuk kelompok meskipun beban komputasi pergerakan berada pada intensitas tinggi.

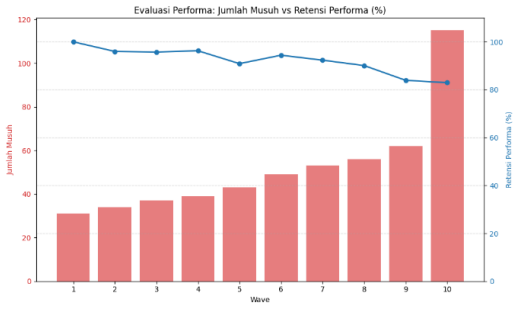
Keseluruhan hasil pengujian visual menegaskan bahwa implementasi algoritma RVO berjalan tepat sesuai dengan rumusan teoretis. Sistem secara konsisten mendeteksi radius ancaman, membagi beban penghindaran sama rata, dan mengkoreksi jalur kecepatan akhir dengan baik. Algoritma RVO berhasil mengatasi masalah tumpang tindih visual (*overlapping*) pada skenario kerumunan musuh tingkat tinggi di dalam *game*.

Evaluasi *Benchmarking*

Tahap evaluasi performa berfokus pada pengujian teknis performa sistem. Parameter utama yang dievaluasi adalah nilai *Frame Per Second* (FPS) pada setiap *wave*. *Game* yang dikembangkan memiliki kemampuan untuk mencatat data jumlah agen musuh yang munul dan rata-rata FPS setiap *wave*. Seluruh data mentah dari 21 *tester* yang telah menguji *game* akan dikumpulkan dan diolah menjadi satu kesatuan data. Pengolahan data menggunakan nilai median untuk mencari angka FPS tengah yang paling representatif dari seluruh variasi perangkat keras pengguna. Data FPS yang didapatkan dari data performa *tester* dan dikonversi menjadi persentase retensi performa, yang bertujuan menghilangkan bias dari perbedaan spesifikasi perangkat keras *tester*. Persentase retensi performa membuktikan seberapa baik sistem menahan penurunan FPS saat jumlah agen musuh bertambah drastis. Hasil pengolahan data ini diekspor ke dalam bentuk tabel rekapitulasi dan divisualisasikan melalui grafik kombinasi sebagai bukti seberapa baik sistem mempertahankan stabilitas *game* seiring bertambahnya kepadatan kerumunan musuh. Hasil rekapitulasi seluruh data *tester* disajikan dalam Tabel 3.

Tabel 3. Tabel Pengolahan Data *Tester*

Wave	*Enemy Count*	*Average* FPS	Retensi Performa (%)
1	31	611,23	100
2	34	586,71	95,99
3	37	585,15	95,73
4	39	588,42	96,27
5	43	555,47	90,88
6	49	577,08	94,41
7	53	564,32	92,33
8	56	550,57	90,08
9	62	512,86	83,91
10	115	506,87	82,93



Gambar 6. Grafik Evaluasi Performa

Tabel 3 menunjukkan tren penurunan performa sistem secara bertahap seiring bertambahnya jumlah musuh pada setiap *wave*. Rekapitulasi data menunjukan jumlah agen musuh yang muncul semakin banyak seiring bertambahnya jumlah *wave*. Pada *wave* 1 sistem memproses rata-rata 31 agen musuh dengan nilai retensi referensi 100%. Beban komputasi terus meningkat hingga

memuncak pada *wave* 10, dimana sistem memproses rata-rata 115 agen musuh dengan retensi performa 82.93%.

Puncak beban komputasi terjadi pada *wave* 10 karena sistem memproses algoritma penghindaran untuk ratusan agen mengkalkulasi algoritma RVO secara bersamaan. Konsistensi pergerakan metrik ini membuktikan perhitungan menggunakan nilai median berhasil menstabilkan variasi data ekstrem.

Gambar 6 memvisualisasikan tren algoritma RVO yang konsisten dan stabil saat menangani agen kerumunan skala besar. Garis berwarna biru yang merepresentasikan retensi performa menampilkan tren penurunan yang landai. Titik terendah retensi performa tercatat pada *wave* 10 dengan nilai 82,93%. Nilai retensi ini bertahan jauh di atas ambang batas toleransi kelayakan performa minimum sebesar 70%. Pengujian ini menghasilkan kesimpulan bahwa sistem terbukti efisien secara komputasi dan mampu mempertahankan kelancaran kinerja yang optimal meskipun memproses rute dan kalkulasi penghindaran tabrakan dari banyak agen musuh.

Nilai FPS dan retensi performa mengalami fluktuasi pada *wave* 5 dan *wave* 6. Rata rata FPS turun menjadi 555,47 pada *wave* 5 lalu kembali naik menjadi 577,08 pada *wave* 6. Fluktuasi ini merupakan akibat dari kemunculan musuh bertipe *boss* pada *wave* 5. Kehadiran entitas *boss* menghadirkan beban komputasi tambahan karena agen *boss* tersebut memiliki radius rintangan yang lebih besar dan logika perilaku yang lebih kompleks.

Evaluasi *User Acceptence Test* (UAT)

Pengujian menggunakan UAT dilakukan untuk mengevaluasi aspek subjektif tester terhadap kualitas visual dan kealamian pergerakan kerumunan musuh di dalam *game*. Pengumpulan data dilakukan dengan mendistribusikan kuesioner kepada target pengguna. Pemilihan responden menggunakan teknik *purposive sampling* yang melibatkan 21 responden. *Purposive sampling* adalah teknik penentuan sampel dengan pertimbangan atau kriteria tertentu yang relevan dengan tujuan penelitian. Kriteria responden pada pengujian ini mencakup mahasiswa aktif Universitas Bunda Mulia angkatan 2022 dari Fakultas Teknologi dan Desain (FTD). Responden wajib berada pada rentang usia 18 hingga 24 tahun dan wajib memiliki pengalaman bermain permainan video (*video game*) menggunakan perangkat *Personal Computer* (PC).

Pertanyaan pada kusioner UAT mengenai kualitas visual adalah sebagai berikut:

1. Apakah karakter musuh terlihat saling menembus fisik/badan satu sama lain (*clipping/overlapping*) saat mereka mengejar Anda dalam kelompok besar?
2. Apakah pergerakan musuh terlihat bergetar (*jittering*) saat mereka berusaha mencari jalan mendekati Anda?

Tabel 4. Tabel Evaluasi Kualitas Visual

Jawaban	Pertanyaan 1	Pertanyaan 2
Selalu	0	0
Sering	0	0
Jarang	6	8
Tidak Pernah	15	13

Berdasarkan data yang diperoleh pada Tabel 4, maka persentase yang didapatkan setiap pertanyaan

$$Persentase = \left(\frac{(15 \times 4) + (6 \times 3)}{(21 \times 4)}\right) \times 100\%$$

$$Persentase = \left(\frac{78}{84}\right) \times 100\% \ = 92.85\%$$

Pertanyaan 1 mengenai masalah tumpang tindih (*overlapping*) agen mendapatkan persentase sebesar 92,85%.

$$Persentase = \left(\frac{(13 \times 4) + (8 \times 3)}{(20 \times 4)}\right) \times 100\%$$
$$Persentase = \left(\frac{76}{84}\right) \times 100\% \ = \ 90.47\%$$

Pertanyaan 2 mengenai masalah getaran pergerakan (*jittering*) mendapatkan persentase sebesar 90,47%.

Hasil perhitungan persentase pada kedua pertanyaan evaluasi masalah visual menunjukkan sentimen yang sangat positif dari para responden. Berdasarkan kriteria interpretasi persentase penerimaan, kedua poin pertanyaan tersebut masuk ke dalam kategori Sangat Baik. Persebaran data menunjukkan tidak ada responden yang memilih opsi Sering atau Selalu pada kedua pertanyaan tersebut. Hal ini membuktikan bahwa algoritma RVO berhasil mencegah agen musuh saling *overlapping* dan *jittering* saat terjadi kepadatan kerumunan.

Evaluasi pergerakan kerumunan yang alami juga dibutuhkan untuk menguji penilaian subjektif responden terhadap perilaku agen yang menggunakan algoritma RVO. Pertanyaan pada kusioner UAT mengenai kealamian pergerakan adalah sebagai berikut:

1. Berikan penilaian skala 1 hingga 5 mengenai tingkat kealamian pergerakan kerumunan musuh dalam *game* ini.

Tabel 5. Tabel Evaluasi Kealamian Pergerakan

Jawaban	Pertanyaan 1
Skala 1	0
Skala 2	0
Skala 3	1
Skala 4	10
Skala 5	10

Berdasarkan data yang diperoleh pada Tabel 5, maka persentase yang didapatkan setiap pertanyaan adalah:

$$Persentase = \left(\frac{(10 \times 5) + (10 \times 4) + (1 \times 3)}{(21 \times 5)}\right) \times 100\%$$
$$Persentase = \left(\frac{93}{105}\right) \times 100\% \ = \ 88.57\%$$

Hasil perhitungan persentase untuk evaluasi tingkat kealamian pergerakan mencapai angka 88.57%. Nilai persentase ini diklasifikasikan ke dalam kategori Sangat Baik pada tabel kriteria interpretasi. Data distribusi jawaban menunjukkan 10 responden memberikan nilai skala lima, dan 10 responden memberikan skala empat. Hanya 1 responden yang memberikan skala tiga, dan tidak ada responden yang memberikan nilai pada skala dua atau satu. Hasil ini mengonfirmasi bahwa arah manuver penghindaran yang dihasilkan oleh algoritma RVO tidak terlihat kaku. Pemain merasa pergerakan kerumunan musuh di dalam *game* terlihat mulus dan menyerupai perilaku kerumunan yang alami di dunia nyata.

Evaluasi UAT menunjukkan tingkat keberhasilan yang tinggi berdasarkan respon dari *tester*. Aspek pencegahan *overlapping*, *jittering*, dan kealamian pergerakan kerumunan seluruhnya mencapai persentase diatas 80%, menandakan tingkat *User Acceptence* berada pada kategori

Sangat Baik. Hasil evaluasi menggunakan UAT membuktikan bahwa implementasi algoritma RVO telah memenuhi standar kenyamanan visual pemain.

SIMPULAN

Penelitian ini membuktikan implementasi algoritma RVO yang dipadukan dengan metode *Direct Seeking* sukses menghasilkan sistem navigasi kerumunan musuh yang realistis dan efisien pada *game roguelike survival* 2D. Algoritma RVO secara efektif memecahkan masalah anomali visual pada pergerakan agen. Data UAT mencatat tingkat keberhasilan pencegahan *overlapping* sebesar 92,85% dan pencegahan *jittering* sebesar 90,47%. *Tester* menilai aspek kealamian pergerakan pada angka 88,57%. Angka persentase ini diklasifikasikan ke dalam kategori Sangat Baik pada tabel kriteria interpretasi dan mengonfirmasi perilaku manuver agen terlihat mulus dan menyerupai kerumunan nyata.

Pengujian teknis melalui metode *Benchmarking* membuktikan algoritma RVO memiliki efisiensi komputasi yang tinggi apabila dioptimasi menggunakan *Spatial Grid*. Sistem mampu mempertahankan retensi performa sebesar 82,93% saat menangani beban pada *wave* 10 yang melibatkan 115 agen musuh. Penelitian ini memberikan solusi atas kelemahan metode pencarian jalur konvensional seperti A* yang mengalami penurunan performa pada kerumunan padat, serta kelemahan metode berbasis gaya seperti *Flocking* yang sering memicu *jittering*. Penerapan algoritma RVO dapat mencegah penurunan performa perangkat keras secara drastis dan memastikan kualitas pengalaman bermain tetap terjaga secara konsisten. Penerapan algoritma RVO yang dioptimasi dengan metode *Spatial Grid* terbukti dapat mempertahankan kelancaran kinerja *game* meskipun memproses kalkulasi algoritma dari banyak agen. Penelitian ini berhasil menjawab kebutuhan akan sistem navigasi cerdas dalam *game* yang mampu menghilangkan *overlapping* dan *jittering* sekaligus menjamin stabilitas kinerja *frame rate* pada perangkat bersumber daya terbatas.

Sistem navigasi kerumunan pada penelitian ini berhasil diimplementasikan dengan persentase keberhasilan tinggi. Namun evaluasi algoritma RVO pada penelitian ini hanya dilakukan pada arena terbuka (*open map*) yang tidak memiliki objek halangan statis. Oleh karena itu, peneliti selanjutnya dapat mengarahkan penelitian melakukan pengujian algoritma *Reciprocal Velocity Obstacles* (RVO) pada skenario peta yang memiliki rintangan kompleks seperti tembok, celah sempit, atau struktur labirin untuk menguji konsistensi manuver penghindarannya pada ruang gerak yang terbatas.

DAFTAR PUSTAKA

[1] L. Himawan and I. G. N. Suryantara, “PENERAPAN GAME EDUKASI TERHADAP BIMBINGAN BELAJAR MATEMATIKA UNTUK KELAS 2 SD,” *Jurnal Algoritma, Logika dan Komputasi*, vol. 4, no. 2, pp. 588–594, Sep. 2023, doi: 10.30813/j-alu.v2i2.4747.

[2] M. L. Haryanti, G. Chardaputeri, K. Wangsa, R. Figo, and R. Indradjadja, “PENERAPAN MULTIMEDIA DEVELOPMENT LIFE CYCLE DALAM MODEL PENGEMBANGAN PERMAINAN VR THE ARCHER,” *Jurnal Algoritma, Logika dan Komputasi*, vol. 8, no. 2, pp. 830–838, 2025, doi: http://dx.doi.org/10.30813/j-alu.v8i2.8223.

[3] E. S. Amrulloh, R. A. Krisdiawan, and I. Lesmana, “Fuzzy-Driven Adaptive NPC Behavior in a Meme-Based Platformer Game for Android Mobile,” *NUANSA INFORMATIKA*, vol. 19, no. 2, pp. 123–132, Jul. 2025, doi: 10.25134/ilkom.v19i2.440.

[4] A. A. Atqia and M. Syani, "IMPLEMENTASI PATHFINDING DENGAN ALGORITMA A* PADA GAME 2D ROGUELIKE SURVIVAL MENGGUNAKAN UNITY AI NAVIGATION SYSTEM," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 13, no. 3S1, Oct. 2025, doi: 10.23960/jitet.v13i3S1.7763.

[5] S. Khan and Z. Deng, "Agent-based Crowd Simulation: An In-depth Survey of Determining Factors for Heterogeneous Behaviour," Jun. 2024. doi: 10.1007/s00371-024-03503-2.

[6] P. Sudkhot, K. W. Wong, and C. Sombattheera, "COLLISION AVOIDANCE AND PATH PLANNING IN CROWD SIMULATION," *ICIC Express Letters*, vol. 17, no. 1, pp. 13–24, Jan. 2023, doi: 10.24507/icicel.17.01.13.

[7] M. Fachri, M. Hariadi, and S. Mardi Susiki Nugroho, "Emotional Reciprocal Velocity Obstacles for Leader-Following: A Velocity Obstacles-Based Steering Behavior," *IEEE Access*, vol. 12, pp. 169977–169987, 2024, doi: 10.1109/ACCESS.2024.3494834.

[8] D. A. Pratama, Y. Mitachul Arif, and F. Nugroho, "Simulasi Kerumunan NPC Pada Game 'Pengenalan Uin Malang' Menggunakan Metode Flocking," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 11, no. 4, pp. 913–928, Aug. 2024, doi: 10.25126/jtiik.1148377.

[9] I. Pratiwi, "Smart Adaptive NPC AI pada Permainan Labirin Menggunakan Algoritma A*," Feb. 2023. doi: https://doi.org/10.47747/jpsii.v4i3.1658.

[10] Y. M. Arif *et al.*, "CSS for CVR: A Reciprocal Velocity Obstacle-based Crowd Simulation System for Non-Playable Character Movement of Campus Virtual Reality," May 2024. doi: http://dx.doi.org/10.62527/joiv.8.2.1997.

[11] B. Ronaldlie and D. Jollyta, "Analisis Kebutuhan Gaming Berdasarkan Prediksi Frame per Second Menggunakan Algoritma Random Forest," 2025. Accessed: Apr. 10, 2026. [Online]. Available: https://www.ejournal.pelitaindonesia.ac.id/index.php/SENATIKA/article/download/5287/2081/19920

[12] A. F. Zakia, "ANALISIS IMPLEMENTASI GAME MOBILE 2D BERBASIS JAVA MENGGUNAKAN EVENT DRIVEN PROGRAMMING STUDI KASUS FLAPPY BIRD," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 14, no. 1, Jan. 2026, doi: 10.23960/jitet.v14i1.8930.

[13] F. A. Simatupang and D. A. Purnamasari, "Optimasi Level of Detail dan Occlusion Culling dalam Game Aaron Lost in the Jungle," *Journal of Applied Multimedia and Networking (JAMN)*, vol. 8, no. 2, 2024, [Online]. Available: http://jurnal.polibatam.ac.id/index.php/JAMN

[14] R. Krisno and D. Bhisetya Rarasati, "PENGEMBANGAN SISTEM APLIKASI PEMBELAJARAN MULTIMEDIA INTERAKTIF UNTUK ANAK BERBASIS ANDROID DEVELOPMENT OF AN ANDROID-BASED INTERACTIVE MULTIMEDIA LEARNING APPLICATION SYSTEM FOR CHILDREN," *Jurnal Algoritma, Logika dan Komputasi*, vol. VI, no. 2, pp. 579–587, 2023, doi: 10.30813/j-alu.v2i2.4742.