Pembangkitan Solusi Labirin dalam Bentuk Citra dengan Simulasi Partikel Acak

Jauhar Wibisono, 13519160
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): jauhar.wibisono@gmail.com

***Abstrak*—Labirin adalah salah satu jenis teka-teki visual. Pemain labirin bertujuan menghubungkan dua buah titik, mulai dan selesai, dengan sebuah kurva kontinu tanpa memotong segmen garis, kurva, atau wilayah yang terlarang. Sebuah metode pembangkit solusi labirin dalam bentuk citra yang memanfaatkan deteksi tepi citra dan simulasi partikel acak dicetuskan. Algoritma metode tersebut dideskripsikan. Algoritma tersebut memiliki kompleksitas waktu kuadratik terhadap besar masukan dan kompleksitas memori linier terhadap panjang solusi. Ide optimasi algoritma tersebut kemudian dipaparkan.**

Kata Kunci—labirin, deteksi tepi, citra, simulasi, partikel

I. PENDAHULUAN

Labirin merupakan salah satu jenis teka-teki visual. Dalam sebuah labirin, pemain diberikan sekumpulan segmen garis atau kurva, titik mulai, dan titik selesai. Pemain lalu diminta mencari solusi labirin, yaitu kurva kontinu yang menghubungkan titik mulai dengan titik selesai tanpa memotong satupun segmen garis atau kurva yang diberikan.

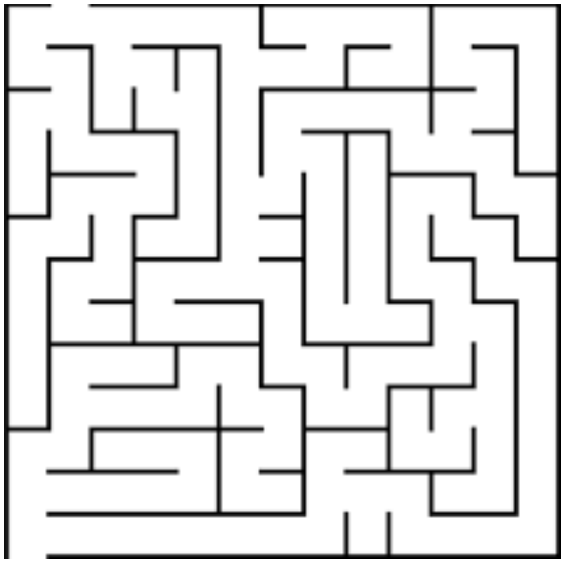

Fig. 1. Contoh labirin, diambil dari [6]

Salah satu cara klasik untuk mencari solusi labirin adalah dengan memodelkan labirin menjadi graf, lalu menerapkan algoritma penelusuran graf seperti DFS atau BFS [1]. Apabila labirin diberikan dalam bentuk citra, metode deteksi tepi atau segmentasi citra dapat digunakan untuk melakukan pemodelan.

Metode deteksi tepi murni untuk melakukan pemodelan, secara khusus, memiliki masalah, yaitu sering meninggalkan lubang-lubang di sepanjang tepi. Lubang tersebut kemudian mungkin dianggap sebagai bagian solusi yang valid. Padahal, melewati lubang tersebut berarti memotong segmen garis atau kurva labirin.

Fig. 2. Citra tepi Lena, terlihat bahwa terdapat lubang-lubang pada kurva-kurva tepi, diambil dari [7]

Salah satu perbedaan wilayah yang valid untuk solusi dengan lubang yang ditimbulkan deteksi tepi adalah ukuran. Lubang yang ditimbulkan deteksi tepi memiliki ukuran yang relatif kecil. Hal ini mendukung ide pembangkitan solusi secara acak untuk menangani adanya lubang hasil deteksi tepi [2]. Algoritma pembangkitan solusi dibuat sedemikian sehingga peluang solusi melewati wilayah yang luas lebih besar daripada

wilayah yang kecil. Tulisan ini membahas tentang penggunaan deteksi tepi dan algoritma pembangkitan solusi acak tersebut.

II. Landasan Teori

A. Deteksi Tepi

Tepi atau *edge* pada citra adalah perubahan nilai intensitas atau nilai keabuan yang besar dalam jarak yang singkat [3]. Secara visual, tepi adalah bagian garis, kurva, atau perubahan wilayah pada citra.

Terdapat banyak metode formal untuk mendeteksi tepi. Metode-metode tersebut memanfaatkan gradien atau perubahan nilai intensitas dan pengambangan atau *thresholding*. Beberapa metode deteksi tepi yaitu Sobel, Prewitt, Roberts, dan Canny.

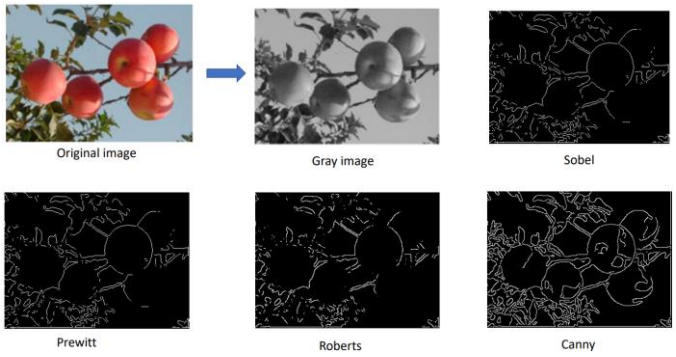

Fig. 3. Contoh hasil berbagai macam metode deteksi tepi, diambil dari [7]

Metode atau operator Canny adalah salah satu operator deteksi tepi. Operator Canny populer karena menghasilkan kumpulan tepi yang memiliki ketebalan satu pixel. Langkah-langkah operator Canny mirip dengan operator deteksi umum pada umumnya, yaitu sebagai berikut.

1. Haluskan citra dengan penapis Gaussian. Penghalusan bertujuan menghilangkan derau yang berpotensi dideteksi sebagai tepi.
2. Hitung magnitudo dan arah gradien menggunakan salah satu matriks kernel deteksi tepi.
3. Kelompokkan pixel menjadi tepi dan bukan tepi dengan melakukan pengambangan atau *thresholding* terhadap nilai magnitudo dan arah gradien.

Salah satu perbedaan operator Canny dengan operator deteksi lainnya adalah pada pengambangan. Operator Canny menggunakan dua nilai ambang sehingga memungkinkan deteksi dua jenis tepi, yaitu tepi kuat dan tepi lemah.

Deteksi tepi merupakan bagian dari analisis citra. Tahapan analisis citra setelah deteksi tepi adalah segmentasi citra, yaitu mereduksi citra menjadi kumpulan wilayah.

Metode deteksi tepi digunakan di tulisan ini sebagai pembangkit lingkungan simulasi partikel. Meskipun segmentasi citra merupakan langkah intuitif dalam pembangkitan solusi labirin, tulisan ini membahas tentang masalah yang muncul pada pembangkitan solusi labirin menggunakan deteksi tepi, sehingga segmentasi citra tidak dipertimbangkan.

B. Simulasi Partikel Acak

Dalam sebuah lingkungan simulasi partikel acak, terdapat dua jenis elemen, yaitu partikel dan tembok [5]. Partikel dapat didefinisikan dengan sebuah titik posisi dan sebuah vektor arah gerak, sedangkan tembok dapat didefinisikan sebagai titik, segmen garis, atau kurva. Dalam lingkungan yang lebih kompleks, partikel dapat memiliki ketebalan atau radius.

Simulasi dilakukan dengan memposisikan partikel-partikel dan tembok-tembok, lalu memberi vektor arah gerak acak ke masing-masing partikel. Partikel-partikel bergerak dalam garis lurus sesuai vektor arah geraknya sampai menabrak tembok. Ketika sebuah partikel menabrak tembok, vektor arah geraknya berubah. Perubahan vektor arah gerak tersebut dapat dibuat mengikuti hukum pemantulan atau secara acak. Dalam lingkungan yang lebih kompleks, tabrakan antara partikel dengan partikel dapat ditinjau.

Metode simulasi partikel acak dalam dua dimensi dan dengan aproksimasi perhitungan digunakan di tulisan ini sebagai metode pembangkitan solusi labirin.

III. Pembahasan

Pembangkitan solusi labirin dalam bentuk citra dengan simulasi partikel acak dilakukan dengan dua tahap, yaitu tahap pemodelan dan tahap simulasi. Masukan dari proses ini adalah citra labirin dan beberapa parameter simulasi, sedangkan keluarannya adalah solusi labirin berupa kumpulan segmen garis yang menghubungkan titik mulai dan titik selesai.

Tahap pertama adalah memodelkan labirin menjadi lingkungan simulasi partikel acak. Lingkungan simulasi partikel dapat dibangkitkan dengan beberapa masukan: citra labirin, wilayah mulai, dan wilayah selesai. Setiap titik atau pixel pada wilayah mulai dijadikan partikel simulasi untuk menyatakan bahwa solusi labirin dimulai dari wilayah tersebut. Metode deteksi tepi diaplikasikan ke citra labirin masukan untuk mendapatkan kumpulan tembok. Terakhir, wilayah selesai digunakan dalam syarat terminasi simulasi. Untuk mempermudah komputasi, partikel, tembok, dan wilayah selesai dapat direpresentasikan sebagai titik latis (titik dengan koordinat bilangan bulat).

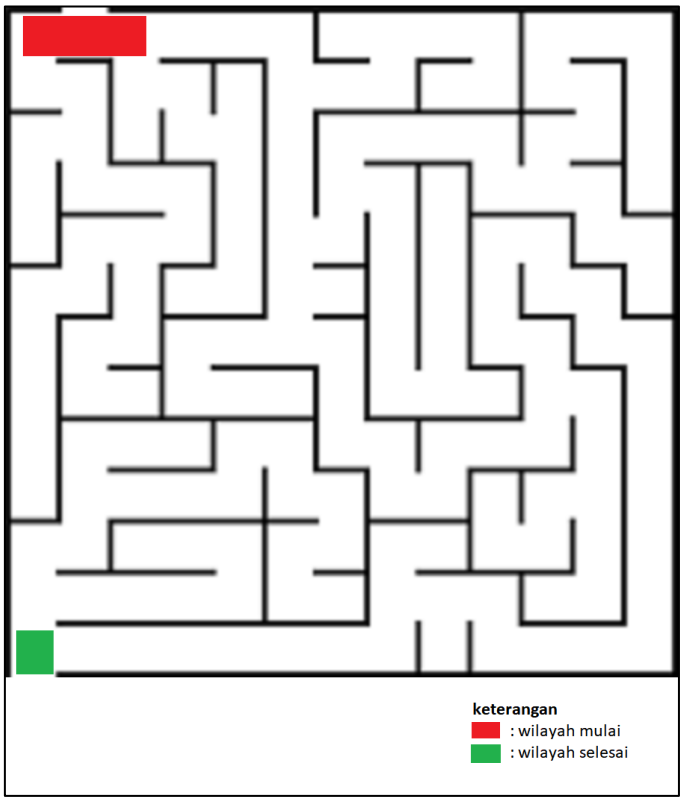

Fig. 4. Contoh masukan untuk model labirin: citra labirin, wilayah mulai, dan wilayah selesai, diambil dari [6] dengan penyesuaian

Tahap kedua adalah mensimulasikan aktivitas partikel. Karya Kolb [5] mendeskripsikan algoritma simulasi partikel secara *real-time*. Untuk keperluan tulisan ini, algoritma tersebut dapat disederhanakan dengan tidak meninjau setiap titik waktu, tetapi meninjau titik waktu yang penting saja, yaitu waktu tabrakan partikel dan waktu terminasi simulasi. Algoritma tersebut adalah seperti berikut.

1. Inisialisasi vektor arah gerak setiap partikel dengan nilai acak.
2. Untuk setiap pasangan partikel - partikel, partikel - tembok, dan partikel – wilayah selesai, hitung waktu tabrakannya. Apabila tidak terjadi tabrakan antara pasangan yang ditinjau, gunakan ∞ atau nilai yang besar sebagai waktu tabrakannya.
3. Cari waktu tabrakan terkecil, misalkan waktu tabrakan tersebut adalah t.
4. Hitung posisi setiap partikel pada waktu t. Untuk partikel-partikel yang mengalami tabrakan pada waktu t, hitung juga vektor arah gerak yang baru.
5. Apabila tabrakan yang terjadi bukan antara partikel dengan wilayah selesai, ulangi algoritma dari langkah 2. Apabila sebaliknya, terminasi simulasi.

Untuk membangkitkan solusi labirin, untuk setiap partikel, titik-titik tabrakannya disimpan. Solusi labirin lalu dibangkitkan dengan menelusuri kumpulan titik tabrakan milik partikel yang pertama kali bertabrakan dengan wilayah selesai. Untuk menangani kasus labirin tanpa solusi, pembatasan iterasi langkah 2 sampai 4 dapat diterapkan.

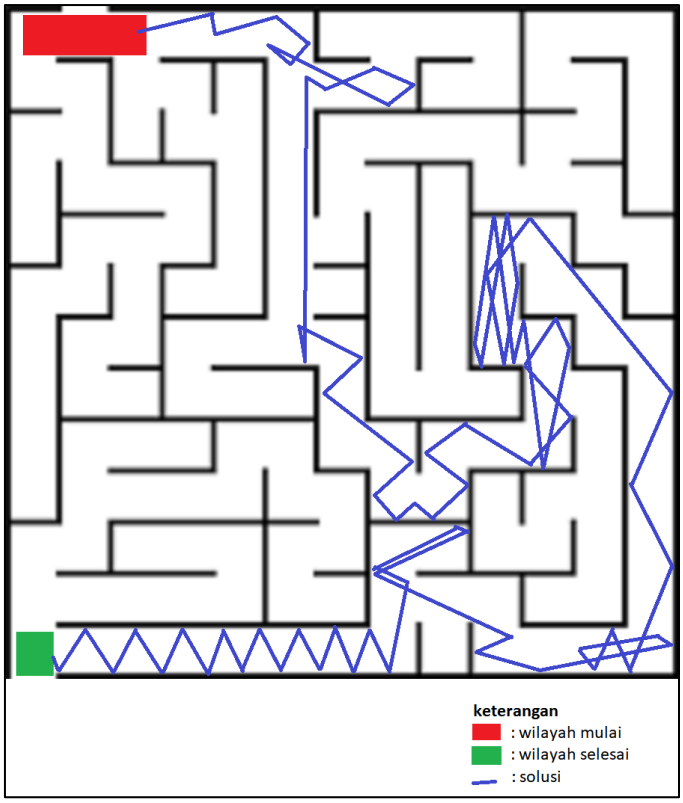

Fig. 5. Ilustrasi solusi hasil pembangkitan yang didapatkan dengan menelusuri kumpulan titik tabrakan sebuah partikel, diambil dari [6] dengan penyesuaian

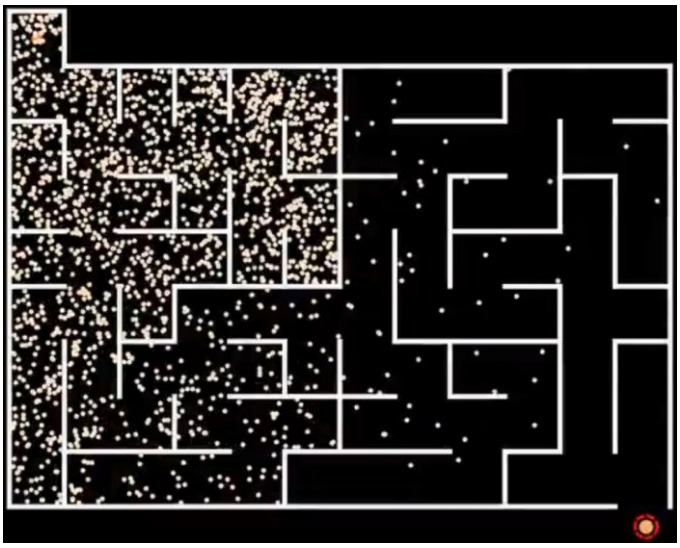

Fig. 6. Contoh keadaan partikel di tengah simulasi, diambil dari [2]

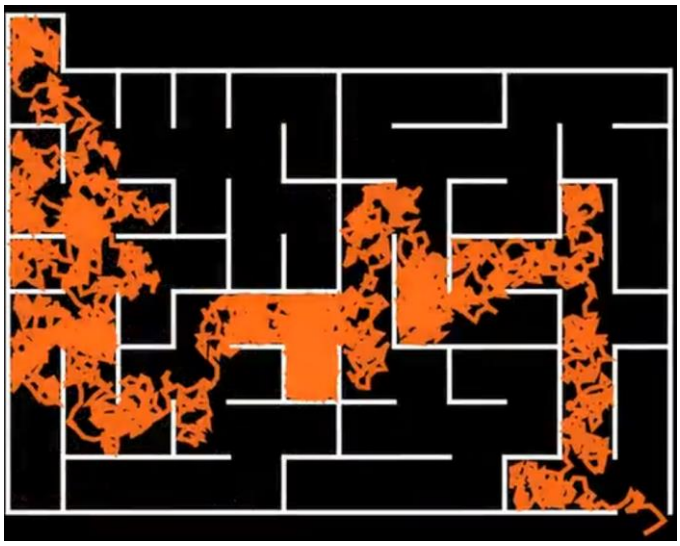

Fig. 7. Contoh solusi hasil pembangkitan dengan simulasi partikel acak, diambil dari [2]

Simulasi partikel di atas diharapkan dapat menyelesaikan masalah lubang pada hasil deteksi tepi citra labirin. Secara intuitif, ukuran lubang pada hasil deteksi tepi yang relatif kecil dibandingkan dengan wilayah yang valid untuk solusi membuat peluang solusi labirin yang dibangkitkan melalui lubang cukup kecil. Intuisi ini dapat dianalogikan dengan penyebaran gas melalui lubang dengan berbagai ukuran.

Penggunaan sumber daya waktu algoritma di atas perlu diperhatikan. Misalkan n_p, n_t, dan n_s menyatakan banyak titik partikel, banyak titik yang menyatakan tembok, dan banyak titik yang menyatakan wilayah selesai. Kompleksitas waktu setiap iterasi langkah 2 sampai 4 algoritma simulasi partikel di atas dapat dinyatakan dengan kompleksitas kuadratik, yaitu $O(n_p^2 + n_p n_t + n_p n_s)$. Kompleksitas ini cukup bagus untuk menyelesaikan kasus yang cukup kecil, seperti kasus dengan luas wilayah mulai sebesar 50 pixel dan citra labirin sebesar 500×500 pixel (setiap iterasi membutuhkan kira-kira seperempat detik).

Selain penggunaan sumber daya waktu, penggunaan sumber daya memori juga perlu diperhatikan. Terdapat bagian algoritma yang membuat penggunaan memori tumbuh terus-menerus, yaitu penyimpanan solusi. Kompleksitas memori algoritma di atas linier terhadap banyak titik pada solusi. Dengan kata lain, semakin panjang solusi labirin masukan, semakin banyak memori yang digunakan.

Beberapa optimasi dapat dilakukan untuk meningkatkan performa algoritma di atas. Pertama, *downsampling* dapat dilakukan terhadap citra labirin masukan untuk mengurangi ukutan labirin. Kekurangan optimasi ini adalah *downsampling* yang dilakukan mungkin menutup beberapa jalur labirin. Kedua, penulisan solusi ke *disk* dapat dilakukan untuk menghindari memori penuh. Terakhir, kompleksitas waktu pencarian waktu tabrakan terkecil pada langkah ketiga algoritma di atas dapat ditingkatkan menggunakan metode *linesweep*. Algoritma dengan metode *linesweep* memiliki kompleksitas waktu kira-kira $O(n \log n)$ dengan n menyatakan banyak titik total (jumlah dari titik partikel, titik yang menyatakan tembok, dan titik yang menyatakan wilayah selesai.

IV. Kesimpulan

Dari pembahasan didapatkan metode pembangkitan solusi labirin dalam bentuk citra dengan simulasi partikel acak yang terdiri atas dua tahap. Metode tersebut memiliki kompleksitas waktu kuadratik terhadap banyak titik (berbagai macam titik) dan kompleksitas memori linier terhadap panjang solusi labirin masukan. Beberapa ide optimasi untuk meningkatkan performa metode pembangkitan solusi tersebut telah dibahas, yaitu *downsampling* citra masukan, penyimpanan menggunakan *disk*, dan pemanfaatan metode *linesweep* untuk meningkatkan kompleksitas waktu.

Tulisan ini tentunya dapat dikembangkan dengan melakukan optimasi yang telah disebutkan di atas. Pengembangan lain yang dapat dilakukan adalah meningkatkan presisi perhitungan dengan menggunakan tipe data *float* dan representasi garis dan kurva. Selain itu, asumsi pada tulisan ini bahwa peluang solusi labirin yang dibangkitkan melewati lubang pada hasil deteksi tepi kecil dapat dibuktikan atau diperkuat.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa,
2. Bapak dosen pengampu mata kuliah Interpretasi dan Pengolahan Citra IF4073,
3. orang tua penulis,
4. asisten mata kuliah Interpretasi dan Pengolahan Citra IF4073,
5. teman-teman penulis, dan
6. pihak-pihak lain

yang telah mendukung penulis selama pembelajaran dan proses penulisan makalah ini.

Referensi

[1] Aqel, M. O., Issa, A., Khdair, M., ElHabbash, M., AbuBaker, M., & Massoud, M. (2017, Oktober). Intelligent maze solving robot based on image processing and graph theory algorithms. In 2017 International Conference on Promising Electronic Technologies (ICPET) (pp. 48-53). IEEE.

[2] Henderson, M (2021, September). https://twitter.com/matthen2/status/1440443280827699206 [diakses 19 Desember 2022].

[3] R. Munir, Pendeteksian Tepi Bagian 1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/18-Pendeteksian-Tepi-Bagian1-2022.pdf [Diakses 19 Desember 2022].

[4] Bhattacharjee, P. R. (2005). The generalized vectorial laws of reflection and refraction. European journal of physics, 26(5), 901.

[5] Kolb, A., Latta, L., & Rezk-Salama, C. (2004, August). Hardware-based simulation and collision detection for large particle systems. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware (pp. 123-131).

[6] publicdomainvectors.org, https://publicdomainvectors.org/id/bebas-vektor/Teka-teki-sederhana-labirin/50724.html [Diakses 19 Desember 2022].

[7] R. Munir, Pendeteksian Tepi Bagian 2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/19-Pendeteksian-Tepi-Bagian2-2022.pdf [Diakses 19 Desember 2022].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Desember 2022

Jauhar Wibisono, 13519160