Penerapan Algoritma Greedy pada Simplifikasi Mesh Tiga Dimensi Menggunakan Edge Collapse Berbasis Quadric Error Metrics

Marcel Luther Sitorus - 13524063
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: sitorusmarcelluther@gmail.com , 135324063@std.stei.itb.ac.id

***Abstract*—**Simplifikasi mesh tiga dimensi diperlukan untuk menurunkan kompleksitas geometri tanpa menghilangkan bentuk utama objek secara berlebihan. Penelitian ini menerapkan strategi algoritma greedy pada simplifikasi mesh segitiga melalui operasi edge collapse berbasis Quadric Error Metrics. Pada setiap iterasi, algoritma memilih edge valid dengan galat kuadratik minimum untuk dikontraksikan. Kontribusi penelitian meliputi implementasi pemilihan greedy menggunakan priority queue, validasi topologi dan geometri, serta pembandingan dengan skema divide and conquer yang mempartisi mesh secara rekursif. Implementasi dikembangkan menggunakan Java 21 dan JavaFX untuk memuat, menampilkan, menyederhanakan, membandingkan, dan mengekspor mesh OBJ. Pengujian terhadap mesh sintetis planar dan bergelombang menunjukkan bahwa greedy QEM dapat mencapai target reduksi secara konsisten, sedangkan pembagian submesh mempercepat pemrosesan pada mesh lebih besar dengan konsekuensi perlindungan boundary dan seam. Hasil ini menunjukkan bahwa greedy merupakan inti pengambilan keputusan simplifikasi, sementara divide and conquer efektif sebagai optimasi skala komputasi.

***Keywords*—**algoritma greedy; simplifikasi mesh; edge collapse; quadric error metrics; divide and conquer; JavaFX

I. PENDAHULUAN

Model tiga dimensi digunakan pada grafika komputer, simulasi, permainan, visualisasi ilmiah, pemindaian objek, manufaktur, dan aplikasi realitas virtual. Kualitas visual yang tinggi umumnya dicapai dengan menambah jumlah vertex dan face. Namun, mesh dengan jutaan segitiga meningkatkan kebutuhan memori, waktu pemuatan, biaya transmisi, serta beban rendering. Ketika objek ditampilkan dari jarak jauh atau pada perangkat terbatas, sebagian besar detail tersebut tidak memberikan peningkatan visual yang sebanding dengan biaya komputasinya. Oleh sebab itu, diperlukan proses simplifikasi yang menurunkan kompleksitas mesh sambil mempertahankan bentuk global dan fitur penting objek [1], [2].

Salah satu metode paling berpengaruh untuk menyelesaikan masalah tersebut adalah Quadric Error Metrics atau QEM yang diperkenalkan oleh Garland dan Heckbert [1]. QEM memperkirakan perubahan geometri akibat penggabungan dua vertex dengan mengakumulasi jarak kuadratik vertex terhadap bidang-bidang face di sekitarnya. Operasi dasar yang digunakan adalah edge collapse, yaitu mengganti dua endpoint suatu edge dengan satu vertex baru dan memperbarui face-face yang terhubung. Operasi ini dilakukan berulang hingga jumlah face mencapai target.

Dari sudut pandang strategi algoritma, proses tersebut memiliki sifat greedy. Pada setiap iterasi, algoritma mengambil kandidat edge dengan biaya QEM terkecil di antara kandidat yang masih valid. Keputusan lokal terbaik itu tidak dibatalkan pada iterasi berikutnya. Setelah collapse dilakukan, hanya lingkungan lokal di sekitar vertex baru yang diperbarui. Strategi ini tidak menjamin solusi global dengan galat minimum untuk seluruh urutan collapse, tetapi menawarkan kompromi yang baik antara kualitas dan efisiensi [3], [4].

Makalah ini tidak hanya menyajikan kajian literatur, melainkan melaporkan implementasi aplikasi simplifikasi mesh yang dikembangkan menggunakan Java 21 dan JavaFX. Program mendukung pembacaan OBJ, visualisasi mesh asli dan hasil simplifikasi, pemilihan target face, eksekusi algoritma di luar JavaFX Application Thread, serta ekspor hasil. Implementasi juga memuat pemeriksaan boundary, segitiga degenerat, pembalikan normal, duplicate face, kondisi link, dan non-manifold edge sebelum suatu kandidat diterima.

Kontribusi penelitian ini terdiri atas tiga bagian. Pertama, algoritma greedy QEM diimplementasikan menggunakan priority queue dan lazy invalidation sehingga edge dengan biaya minimum dapat dipilih secara sistematis. Kedua, validasi topologi dan geometri diterapkan untuk menjaga validitas hasil. Ketiga, dikembangkan mode divide and conquer sebagai perluasan yang membagi mesh menurut centroid face pada sumbu bounding box terpanjang, menjalankan greedy QEM pada setiap submesh, lalu menggabungkan hasil dengan perlindungan seam. Fokus klasifikasi makalah tetap algoritma greedy karena keputusan collapse pada seluruh leaf maupun tahap refinement tetap ditentukan oleh biaya minimum.

Tujuan eksperimen adalah mengevaluasi ketercapaian target jumlah face, waktu eksekusi, serta pengaruh partisi rekursif terhadap proses simplifikasi. Pengujian menggunakan mesh sintetis planar dan bergelombang agar ukuran, konektivitas, dan kompleksitas geometri dapat dikontrol. Pembahasan hasil menekankan trade-off antara keputusan greedy global dan penyederhanaan lokal pada submesh.

II. LANDASAN TEORI

A. Representasi Mesh Segitiga

Mesh segitiga direpresentasikan sebagai himpunan vertex V, edge E, dan face F. Setiap vertex memiliki posisi tiga dimensi, setiap face menghubungkan tiga vertex, dan setiap edge menyatakan pasangan vertex yang muncul bersama pada satu atau dua face. Edge yang hanya dimiliki satu face disebut boundary edge, sedangkan edge interior pada mesh manifold umumnya dimiliki tepat dua face. Struktur

adjacency diperlukan agar algoritma dapat menemukan face dan edge yang terdampak oleh suatu collapse.

Implementasi menggunakan objek MeshVertex, MeshEdge, MeshFace, dan MeshModel. Identitas vertex dan face disimpan secara stabil melalui bilangan bulat. MeshModel membangun peta adjacency dan edge berdasarkan pasangan identitas vertex. Pendekatan ini bukan half-edge lengkap, tetapi menyediakan operasi yang dibutuhkan untuk pencarian tetangga, pendeteksian boundary, penghapusan face, dan pembentukan ulang topologi setelah collapse.

B. Operasi Edge Collapse

Pada edge collapse, edge e = (v1, v2) diganti oleh satu vertex baru vbar. Semua face yang hanya menggunakan salah satu endpoint diperbarui agar menggunakan vbar. Face yang sebelumnya menggunakan kedua endpoint menjadi degenerat dan dihapus. Operasi ini mengurangi sedikitnya satu vertex dan biasanya dua face pada edge interior. Karena efeknya lokal, edge collapse cocok digunakan untuk membentuk level of detail dan progressive mesh [2].

Meskipun tampak sederhana, collapse dapat menghasilkan geometri tidak valid. Contohnya adalah face dengan dua vertex sama, face berluas mendekati nol, normal yang berbalik, edge yang digunakan lebih dari dua face, serta vertex fan yang terputus. Oleh karena itu, edge dengan biaya terkecil hanya boleh diterima setelah melewati pemeriksaan topologi dan geometri.

C. Quadric Error Metrics

Untuk setiap face, dihitung persamaan bidang ternormalisasi ax + by + cz + d = 0. Koefisien bidang dituliskan sebagai vektor homogen p = [a b c d]^T. Matriks quadric face dibentuk dengan perkalian luar vektor tersebut.

$$Kp = p\ p^T \quad (1)$$

Quadric setiap vertex merupakan jumlah quadric seluruh face yang bersebelahan dengan vertex tersebut.

$$Qv = \Sigma\ Kp \quad (2)$$

Untuk edge e = (v1, v2), quadric gabungannya diperoleh dengan menjumlahkan quadric kedua endpoint.

$$Qe = Qv1 + Qv2 \quad (3)$$

Jika posisi pengganti homogen adalah vbar = [x y z 1]^T, biaya collapse dihitung sebagai berikut.

$$\text{cost}(e, vbar) = vbar^T\ Qe\ vbar \quad (4)$$

Posisi optimal diperoleh dengan meminimalkan fungsi kuadratik tersebut. Implementasi menyelesaikan sistem linier dari bagian kiri atas matriks Qe. Jika sistem singular atau tidak stabil secara numerik, kandidat posisi v1, v2, dan titik tengah dievaluasi secara langsung. Kandidat dengan biaya terkecil digunakan. Mekanisme fallback penting pada permukaan planar karena sistem matriks sering tidak memiliki invers unik.

Biaya QEM mengukur seberapa jauh vertex baru menyimpang dari kumpulan bidang lokal, bukan jarak Euclidean biasa antara endpoint. Dengan demikian, edge yang pendek belum tentu selalu menjadi pilihan terbaik; edge pada daerah datar biasanya memiliki biaya rendah, sedangkan edge di sekitar fitur tajam cenderung mempunyai biaya lebih tinggi [1], [5].

D. Strategi Greedy

Algoritma greedy membangun solusi melalui rangkaian keputusan lokal. Dalam konteks ini, himpunan kandidat berisi edge aktif beserta biaya collapse-nya. Pada setiap iterasi dipilih edge valid dengan biaya minimum:

$$e^* = \arg\min(\text{cost}(e)), \text{untuk } e \in \text{Evalid} \quad (5)$$

Kandidat disimpan dalam min-priority queue. Ketika biaya atau topologi edge berubah, entri lama tidak dihapus secara langsung. Setiap vertex memiliki nomor versi dan kandidat menyimpan snapshot versi kedua endpoint. Saat kandidat dikeluarkan dari antrean, kandidat dianggap stale apabila edge telah hilang, endpoint tidak aktif, atau versinya tidak cocok. Teknik lazy invalidation menghindari kebutuhan untuk mencari dan menghapus entri tertentu dari heap.

Setelah collapse diterima, quadric dan kandidat edge pada lingkungan vertex hasil collapse dihitung kembali. Implementasi proyek membangun ulang quadric setelah collapse untuk menjaga konsistensi, kemudian memasukkan kembali kandidat edge tetangga. Walaupun langkah ini lebih konservatif daripada pembaruan quadric yang sepenuhnya lokal, prioritas global tetap dikelola oleh antrean minimum.

E. Divide and Conquer sebagai Perluasan

Divide and conquer memecah masalah menjadi submasalah, menyelesaikan submasalah secara rekursif, lalu menggabungkan hasil. Pada implementasi, mesh dibagi menggunakan median centroid face di sepanjang sumbu bounding box terpanjang. Setiap anak memperoleh subset face dan salinan vertex yang diperlukan. Vertex yang digunakan oleh kedua anak ditandai sebagai shared boundary berdasarkan original vertex ID.

Tahap conquer tetap menggunakan GreedyQemSimplifier. Vertex pada batas partisi dilindungi agar simplifikasi dua anak tidak menghasilkan konektivitas berbeda dan menimbulkan crack. Tahap combine menggunakan MeshMerger untuk menyatukan vertex dengan original ID yang sama, menghapus face duplikat, serta membangun ulang topologi. Bila jumlah face masih di atas target, seam refinement dapat menjalankan greedy QEM lagi pada hasil gabungan. Karena itu, divide and conquer pada penelitian ini merupakan organisasi komputasi di tingkat global, sedangkan greedy tetap menjadi strategi keputusan pada setiap leaf.

III. METODOLOGI DAN IMPLEMENTASI

A. Alur Sistem

Alur aplikasi dimulai ketika pengguna memuat berkas Wavefront OBJ. ObjMeshReader membaca baris vertex dan face, mendukung indeks positif, indeks relatif negatif, serta token face dalam bentuk v, v/vt, v//vn, dan v/vt/vn. Polygon sederhana ditriangulasi dengan fan triangulation. Setelah proses parsing, MeshValidator memeriksa referensi vertex, face degenerat, duplicate face, boundary, connected component, dan non-manifold edge.

Mesh asli kemudian dikonversi menjadi JavaFX TriangleMesh dan ditampilkan melalui MeshViewport. Pengguna dapat memilih mode GREEDY_QEM atau DIVIDE_AND_CONQUER_QEM, mengatur target face, persentase reduksi, threshold normal, boundary preservation, leaf face threshold, kedalaman rekursi, pemrosesan paralel, dan seam refinement. Proses simplifikasi dijalankan

menggunakan JavaFX Task agar antarmuka tetap responsif. Setelah selesai, hasil ditampilkan berdampingan dan dapat diekspor kembali sebagai OBJ.

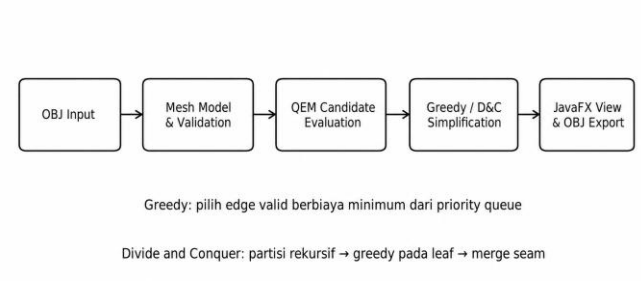


Fig. 1. Alur implementasi aplikasi simplifikasi mesh.

B. Tahapan Greedy QEM

1) Hitung statistik dan salin elemen aktif dari mesh masukan.
2) Hitung bidang setiap face serta quadric awal seluruh vertex.
3) Bentuk kandidat untuk setiap edge dan masukkan ke priority queue.
4) Ambil kandidat dengan biaya minimum.
5) Tolak kandidat stale atau kandidat yang tidak lolos validasi.
6) Lakukan collapse terhadap kandidat valid.
7) Hitung kembali quadric dan masukkan kandidat baru di sekitar vertex yang dipertahankan.
8) Ulangi hingga target tercapai, antrean kosong, atau operasi dibatalkan.

Kompleksitas inisialisasi quadric dan kandidat bersifat linear terhadap jumlah face dan edge. Pengambilan kandidat minimum membutuhkan O(log |E|). Jumlah iterasi sebanding dengan jumlah collapse. Namun, biaya aktual juga dipengaruhi oleh validasi dan pembangunan ulang sebagian struktur. Pada mesh manifold segitiga, |E| secara umum berorde sama dengan |F| sehingga priority queue tetap memberikan struktur pemilihan yang sesuai untuk strategi greedy.

C. Validasi Kandidat

CollapseValidator menerapkan serangkaian pemeriksaan. Pertama, koordinat kandidat harus finite dan biaya tidak boleh NaN. Kedua, aturan boundary mencegah penggabungan vertex boundary dengan vertex interior secara sembarang. Ketiga, protected original vertex ID mencegah perubahan seam pada leaf divide and conquer. Keempat, link condition membandingkan irisan tetangga endpoint dengan vertex lawan pada face yang menggunakan edge.

Face terdampak disimulasikan dengan mengganti endpoint yang dihapus menjadi vertex yang dipertahankan. Face yang berubah menjadi degenerat akan dihapus, tetapi kandidat ditolak apabila menghasilkan duplicate face, area mendekati nol pada face yang seharusnya bertahan, atau normal baru dengan dot product di bawah threshold. Pemeriksaan tambahan memastikan tidak ada edge baru yang digunakan lebih dari dua face. Pendekatan konservatif ini dapat menyebabkan proses berhenti sebelum target, tetapi lebih aman daripada menghasilkan mesh rusak.

D. Tahapan Divide and Conquer

Fungsi rekursif menerima mesh, target lokal, kedalaman, dan konfigurasi. Base case terjadi ketika jumlah face di bawah leaf threshold, kedalaman maksimum tercapai, ukuran mesh hampir sama dengan target, atau pembatalan diminta. Pada base case, greedy QEM dijalankan langsung.

Pada tahap divide, MeshPartitioner menentukan sumbu bounding box terpanjang, mengurutkan centroid face, dan membagi face di sekitar median. Target global dibagi proporsional terhadap jumlah face anak. Shared original vertex ID ditambahkan ke protected set. Bila pemrosesan paralel aktif, dua anak dieksekusi melalui ForkJoinPool. Setelah kedua hasil tersedia, MeshMerger menggabungkan mesh berdasarkan identitas asli.

Mode ini memperkecil ukuran priority queue pada setiap leaf dan memungkinkan paralelisme. Akan tetapi, keputusan edge pada satu partisi tidak dibandingkan langsung dengan edge pada partisi lain. Akibatnya, urutan collapse berbeda dari greedy global dan hasil akhir dapat berbeda meskipun target sama. Perlindungan boundary juga dapat mengurangi ruang solusi lokal.

E. Parameter Pengujian

Pengujian dilakukan terhadap empat mesh sintetis: grid 10×10, grid 20×20, wavy grid 20×20, dan wavy grid 30×30. Grid planar memiliki z konstan, sedangkan wavy grid menggunakan variasi tinggi sinusoidal sehingga menghasilkan perubahan normal dan kurvatur lokal. Setiap sel grid dibagi menjadi dua segitiga. Target ditetapkan sebesar 40 persen dari face awal atau reduksi 60 persen.

Parameter boundary preservation diaktifkan, epsilon numerik menggunakan nilai kecil positif, dan normal flip threshold mempertahankan orientasi face. Untuk divide and conquer digunakan partisi centroid median, protected shared boundary, serta seam refinement. Waktu yang dilaporkan adalah durasi satu eksekusi pada lingkungan pengujian proyek. Karena belum dilakukan warm-up JVM dan pengulangan statistik yang panjang, angka waktu diperlakukan sebagai indikasi komparatif, bukan benchmark absolut.

IV. HASIL DAN PEMBAHASAN

A. Hasil Kuantitatif

Tabel I memperlihatkan jumlah face awal, target, face akhir, dan waktu eksekusi. Nilai waktu berasal dari pengujian implementasi Java 21 pada mesh sintetis yang dibentuk secara deterministik.

Objek uji	Face awal	Target	Greedy akhir	Greedy (ms)	D&C akhir	D&C (ms)
Grid 10×10	200	80	80	3320	80	312
Grid 20×20	800	320	320	6734	320	2080
Wavy 20×20	800	320	320	3399	320	876
Wavy 30×30	1800	720	720	10725	712	1132

TABEL I. HASIL EKSPERIMEN SIMPLIFIKASI

Greedy QEM mencapai target pada seluruh kasus uji. Hasil ini menunjukkan bahwa masih tersedia cukup banyak kandidat aman meskipun boundary dan normal dilindungi. Jumlah face akhir sama dengan target, sehingga strategi berhenti berdasarkan kriteria target, bukan karena antrean kandidat habis. Pada grid planar, biaya QEM banyak edge bernilai sangat rendah karena seluruh face berada pada bidang yang sama. Pemilihan greedy kemudian terutama dipengaruhi oleh constraint boundary dan topology.

Pada wavy grid, bidang face tidak koplanar sehingga biaya collapse lebih bervariasi. Edge pada area yang relatif datar memperoleh prioritas lebih tinggi, sedangkan edge dengan perubahan normal lebih besar cenderung dipertahankan. Sifat tersebut sesuai dengan tujuan QEM, yaitu mempertahankan fitur yang memberikan kontribusi besar terhadap galat bidang lokal.

Mode divide and conquer mencapai target pada tiga kasus pertama dan menghasilkan 712 face pada target 720 untuk wavy grid 30×30. Nilai yang sedikit lebih rendah dapat terjadi karena collapse pada leaf menghapus dua face sekaligus dan pembagian target integer tidak selalu menghasilkan jumlah yang tepat setelah merge. Seam refinement juga dapat melakukan collapse tambahan ketika hasil gabungan masih berada di atas anggaran. Perbedaan ini perlu dilaporkan karena menunjukkan bahwa target face pada edge collapse sering bersifat batas, bukan selalu angka yang dapat dicapai persis.

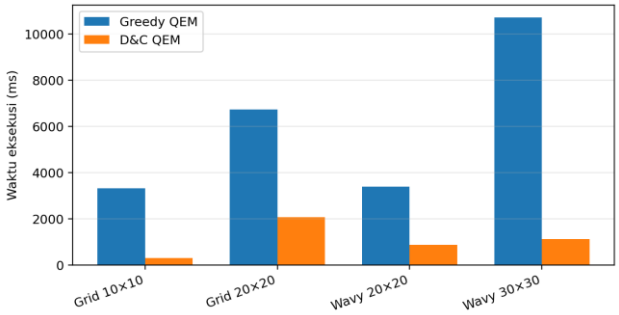


Fig. 2. Perbandingan waktu eksekusi pada data uji.

B. Analisis Waktu Eksekusi

Pada data pengujian, divide and conquer lebih cepat daripada greedy global. Peningkatan paling menonjol terlihat pada wavy grid 30×30, dari 10725 ms menjadi 1132 ms. Ada beberapa penyebab. Pertama, setiap leaf memiliki priority queue lebih kecil. Kedua, validasi dilakukan pada lingkungan submesh yang lebih kecil. Ketiga, ketika paralelisme aktif, anak rekursif dapat diproses secara bersamaan.

Walaupun demikian, angka tersebut tidak boleh dianggap sebagai bukti umum bahwa divide and conquer selalu lebih cepat. Implementasi greedy menghitung kembali quadric setelah setiap collapse, sehingga biaya per iterasi relatif besar. Selain itu, pengukuran satu kali dipengaruhi warm-up JIT, garbage collection, dan beban sistem. Eksperimen lanjutan seharusnya menjalankan beberapa warm-up dan minimal puluhan pengulangan, kemudian melaporkan median serta simpangan baku.

Divide and conquer juga menambah biaya divide, copy submesh, perlindungan seam, dan merge. Pada mesh kecil, overhead tersebut dapat mendominasi. Oleh sebab itu, leafFaceThreshold menjadi parameter penting. Nilai terlalu kecil menghasilkan banyak partisi dan biaya penggabungan tinggi, sedangkan nilai terlalu besar mengurangi manfaat partisi.

C. Kualitas dan Validitas Mesh

Kualitas simplifikasi tidak cukup dinilai dari jumlah face. Mesh hasil juga harus tetap dapat dirender, tidak mengandung indeks invalid, NaN, face berluas nol, duplicate face, atau edge non-manifold. Implementasi menjalankan validasi sesudah load, selama evaluasi collapse, setelah merge, sebelum render, dan sebelum ekspor. Test unit mencakup matematika plane dan quadric, singular fallback, boundary detection, collapse valid, duplicate prevention, pelaporan non-manifold, cancellation, partitioning, merge, serta round-trip OBJ.

Boundary preservation mengurangi risiko perubahan siluet pada mesh terbuka. Namun, aturan konservatif juga dapat mempertahankan terlalu banyak vertex di sekitar batas. Pada objek dengan banyak lubang atau partisi, target agresif mungkin tidak tercapai. Ini merupakan trade-off yang wajar: mempertahankan validitas dan kontinuitas lebih penting daripada memaksakan jumlah face tertentu.

Normal flip threshold berfungsi sebagai batas distorsi lokal. Threshold yang terlalu rendah mengizinkan perubahan bentuk besar, sedangkan threshold terlalu tinggi dapat menolak hampir semua collapse di daerah melengkung. Dengan demikian, konfigurasi optimal bergantung pada karakteristik model dan tujuan visual.

D. Evaluasi Kontribusi Strategi Algoritma

Kontribusi utama dari sudut pandang IF2211 adalah penerapan greedy choice secara eksplisit dan terukur. Fungsi biaya QEM memberikan urutan preferensi kandidat. Priority queue merealisasikan pemilihan minimum. Validator mendefinisikan himpunan kandidat yang feasible. Setelah sebuah edge dikontraksikan, keputusan tersebut bersifat irreversible dan keadaan mesh berubah. Struktur ini memenuhi ciri utama algoritma greedy.

Divide and conquer tidak menggantikan greedy choice. Metode tersebut hanya mengubah ruang tempat greedy dijalankan: kandidat dibandingkan di dalam leaf, bukan selalu terhadap semua edge global. Hasil eksperimen menunjukkan manfaat performa sekaligus kemungkinan perbedaan hasil. Hal tersebut menjadi temuan penting karena menunjukkan batas optimalitas lokal. Greedy global memiliki pilihan kandidat yang lebih luas pada setiap iterasi, sementara greedy lokal pada partisi lebih mudah diparalelkan dan lebih hemat struktur data.

Dengan demikian, sistem dapat dipandang sebagai dua tingkat strategi. Tingkat pertama adalah organisasi divide, conquer, dan combine. Tingkat kedua adalah pemilihan edge greedy berbasis QEM. Untuk memenuhi klasifikasi tugas, makalah menempatkan algoritma greedy sebagai fokus karena seluruh operasi penyederhanaan aktual dilakukan oleh GreedyQemSimplifier.

E. Keterbatasan

Implementasi memiliki beberapa keterbatasan. Struktur mutable topology belum menggunakan half-edge penuh sehingga sebagian pembaruan dilakukan dengan membangun ulang peta adjacency. Atribut OBJ seperti texture coordinate, normal file, material, dan grup tidak dipertahankan pada ekspor. Fan triangulation mengasumsikan polygon sederhana dan tidak menjamin hasil benar untuk polygon konkaf kompleks.

Pengujian utama masih menggunakan mesh sintetis. Pengukuran kualitas belum memakai Hausdorff distance, normal deviation rata-rata, volume preservation, atau perceptual metric. Data waktu juga belum berupa benchmark statistik berulang. Selain itu, pembagian target divide and conquer masih proporsional terhadap jumlah

face, belum mempertimbangkan curvature, luas permukaan, atau kepadatan fitur.

Keterbatasan tersebut tidak menghilangkan kontribusi implementasi, tetapi menentukan ruang interpretasi hasil. Sistem sudah cukup untuk memperlihatkan penerapan strategi greedy dan divide and conquer, namun penelitian lanjutan diperlukan untuk menyimpulkan kualitas pada dataset mesh dunia nyata.

V. KESIMPULAN

Penelitian ini berhasil menerapkan algoritma greedy untuk simplifikasi mesh tiga dimensi menggunakan Edge Collapse berbasis Quadric Error Metrics. Setiap iterasi memilih kandidat edge valid dengan biaya minimum dari priority queue, kemudian memperbarui topologi dan kandidat di sekitarnya. Validasi boundary, link condition, degenerasi face, duplicate face, non-manifold edge, koordinat finite, dan pembalikan normal menjaga agar reduksi tidak merusak mesh.

Implementasi Java 21 dan JavaFX membentuk aplikasi lengkap untuk memuat OBJ, menampilkan mesh asli dan hasil simplifikasi, menjalankan algoritma secara asynchronous, membandingkan statistik, dan mengekspor hasil. Eksperimen pada grid planar dan bergelombang menunjukkan bahwa greedy QEM mencapai target face secara konsisten. Mode divide and conquer memberikan waktu yang lebih rendah pada data uji dengan mempartisi mesh, menjalankan greedy pada leaf, melindungi shared boundary, dan menggabungkan hasil.

Hasil tersebut menegaskan bahwa greedy merupakan strategi inti pengambilan keputusan, sedangkan divide and conquer merupakan optimasi organisasi komputasi. Pengembangan berikutnya dapat menambahkan half-edge penuh, preservasi sharp feature dan atribut, target berbobot curvature, metrik galat geometri, serta benchmark berulang pada dataset nyata.

APENDIKS A. PROTOKOL PENGUJIAN

Apendiks ini merangkum prosedur yang digunakan agar pengujian dapat direplikasi tanpa menyertakan kode program. Setiap mesh dibentuk secara deterministik sebagai kisi persegi. Grid berukuran $n \times n$ memiliki $(n+1)^2$ vertex dan $2n^2$ face. Pada wavy grid, koordinat tinggi dihitung menggunakan kombinasi fungsi sinus dan kosinus sehingga bidang-bidang lokal tidak seluruhnya koplanar. Semua face menggunakan orientasi winding yang konsisten.

Sebelum simplifikasi, validator memastikan tidak ada indeks di luar rentang, face duplikat, koordinat non-finite, atau edge yang dipakai lebih dari dua face. Target reduksi ditentukan dari jumlah face aktif. Waktu dimulai setelah mesh tersedia di memori dan dihentikan saat SimplificationResult terbentuk. Waktu pemuatan OBJ, pembuatan viewport, dan ekspor tidak dimasukkan ke durasi algoritma.

Untuk mode greedy, antrean kandidat dibentuk dari seluruh edge aktif. Untuk mode divide and conquer, leaf threshold dan kedalaman maksimum membatasi rekursi, shared boundary dilindungi, dan hasil anak digabungkan sebelum refinement. Keberhasilan eksperimen ditentukan oleh empat indikator: mesh hasil dapat divalidasi, jumlah face tidak melebihi target secara tidak terkendali, tidak ada koordinat NaN atau tak hingga, dan hasil dapat dikonversi menjadi JavaFX TriangleMesh serta ditulis kembali sebagai OBJ.

A. Kasus Uji Otomatis

Pengujian matematika memverifikasi normalisasi bidang, pembentukan outer product, penjumlahan quadric, evaluasi biaya, penyelesaian posisi optimum, dan fallback ketika matriks singular. Kasus singular penting karena permukaan planar menghasilkan banyak konfigurasi yang tidak memiliki solusi invers tunggal.

Pengujian topologi mencakup satu segitiga, dua segitiga pembentuk quad, tetrahedron, grid terbuka, dan contoh non-manifold. Pemeriksaan memastikan boundary edge terdeteksi, collapse valid mengurangi face, face degenerat dihapus, duplicate face tidak terbentuk, dan link condition menolak perubahan konektivitas yang tidak sah.

Pengujian algoritma memastikan kandidat minimum yang masih fresh diproses lebih dahulu, cancellation menghentikan iterasi, dan proses berhenti ketika target tercapai atau tidak ada collapse aman. Pengujian divide and conquer memeriksa pemilihan sumbu split, keseimbangan partisi, shared vertex ID, base case rekursif, merge, dan validitas hasil akhir.

Pengujian I/O menggunakan variasi token OBJ, polygon quad, indeks negatif, indeks invalid, dan proses export-reload. Hasil ekspor hanya memuat vertex dan face aktif dengan indeks yang dipadatkan. Dengan demikian, file hasil dapat dibaca kembali tanpa mengandalkan identitas internal mesh.

B. Ancaman terhadap Validitas

Validitas internal dipengaruhi oleh kondisi runtime Java. Just-in-time compilation dapat membuat eksekusi awal lebih lambat daripada eksekusi berikutnya. Garbage collection dan proses sistem lain juga dapat mengubah waktu. Karena pengujian yang dilaporkan belum berupa microbenchmark terkontrol, perbandingan waktu digunakan untuk menunjukkan kecenderungan, bukan rasio percepatan universal.

Validitas eksternal dibatasi oleh jenis mesh. Grid sintetis memiliki konektivitas teratur dan tidak mewakili seluruh model hasil pemindaian atau CAD. Mesh nyata dapat mengandung fitur tajam, komponen terpisah, lubang, face sangat kecil, polygon non-konveks, serta atribut tekstur. Oleh sebab itu, hasil pada grid tidak otomatis berlaku pada seluruh dataset.

Validitas konstruk dipengaruhi oleh metrik evaluasi. Jumlah face dan waktu tidak langsung menyatakan kualitas visual. Evaluasi berikutnya perlu menghitung jarak Hausdorff, normal deviation, perubahan volume, dan persepsi pengguna. Meski demikian, QEM sendiri tetap memberikan indikator galat lokal yang digunakan secara konsisten oleh algoritma.

LINK YOUTUBE

https://drive.google.com/drive/folders/1jFmQrWgENCdlNrGsARZWInWzq5zAY6ll?usp=drive_link

LINK REPOSITORY

LutherCeltors/mesh-simplification-algorithm

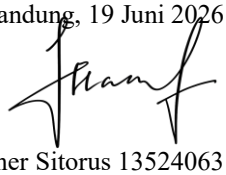
ACKNOWLEDGMENT

Penulis mengucapkan terima kasih kepada mata kuliah IF2211 Strategi Algoritma yang memberikan ruang untuk menerapkan paradigma algoritma pada persoalan komputasi geometri.

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 Juni 2026



Marcel Luther Sitorus 13524063

REFERENCES

[1] M. Garland and P. S. Heckbert, “Surface simplification using quadric error metrics,” in Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, 1997, pp. 209–216.

[2] H. Hoppe, “Progressive meshes,” in Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, 1996, pp. 99–108.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 4th ed. Cambridge, MA: MIT Press, 2022.

[4] A. Levitin, Introduction to the Design and Analysis of Algorithms, 3rd ed. Boston, MA: Pearson, 2012.

[5] M. Garland, Quadric-Based Polygonal Surface Simplification, Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 1999.

[6] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, Polygon Mesh Processing. Natick, MA: A K Peters, 2010.

[7] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, “Interactive multi-resolution modeling on arbitrary meshes,” in Proceedings of SIGGRAPH, 1998, pp. 105–114.

[8] P. S. Heckbert and M. Garland, “Survey of polygonal surface simplification algorithms,” Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., 1997.

[9] H.-T. D. Liu, M. Gillespie, B. Chislett, N. Sharp, A. Jacobson, and K. Crane, “Surface simplification using intrinsic error metrics,” ACM Transactions on Graphics, 2023.

[10] H.-T. D. Liu, X. Zhang, and C. Yuksel, “Simplifying triangle meshes in the wild,” ACM Transactions on Graphics, 2024.