KATTA HAJMLI SAHNALARNI 3D GAUSSIAN SPLATTING ASOSIDA REAL VAQTDA VIZUALLASHTIRISHDA XOTIRA VA HISOBLASH RESURSLARINI OPTIMALLASHTIRISH

Abdubannoyeva Muxlisaxon Iqboljon qizi
Qo'qon davlat Universiteti
Ta'limda axborot texnologiyalari yo'nalishi magistranti
muxlisaxonabdubannoyeva@gmail.com
https://orcid.org/0009-0000-9073-315X
Tel:998-90-058-88-78
https://doi.org/10.5281/zenodo.22344437

ANNOTATSIYA. 3D Gaussian Splatting (3DGS) real vaqtda yuqori sifatli yangi ko'rinishlarni sintez qilishda inqilob qilgan bo'lsa-da, uni katta hajmli sahnalarga (masalan, shahar ko'chalari, me'moriy majmualar, aerofotosuratlar asosidagi hududlar) qo'llashda Gauss primitivlari sonining keskin ortishi xotira va hisoblash resurslari muammosini keltirib chiqaradi. Zamonaviy GPU xotirasi (odatda 24GB) katta hajmli sahnalarni yuqori aniqlikda o'qitish va render qilish uchun yetarli emas. Ushbu tezisda katta hajmli meros obyektlarini real vaqtda vizuallashtirishda resurslarni optimallashtirishning kompleks strategiyasi taklif etiladi. Strategiya uchta asosiy komponentdan iborat: (1) kameraga masofaga asoslangan Level-of-Detail (LOD) usuli – optimal Gauss to'plamlarini iterativ tanlash orqali render vaqti va GPU xotirasini kamaytiradi; (2) visibility-in-block bo'linish strategiyasi – sahna bloklarga bo'linib, faqat ko'rinadigan piksellar o'qitishga jalb qilinadi; (3) host offloading mexanizmi – Gauss primitivlari host xotirasida saqlanib, faqat kerakli qismi GPUga yuklanadi.

Kalit so'zlar: 3D Gaussian Splatting, katta hajmli sahnalar, Level-of-Detail, xotira optimallashtirish, real vaqtda vizuallashtirish

3D Gaussian Splatting (3DGS) so'nggi ikki yil ichida radiance field rekonstruksiyasi sohasida eng muhim yutuqlardan biriga aylandi. Uning NeRFga nisbatan asosiy afzalligi – real vaqtda render qilish tezligi va yuqori vizual sifatdir. Biroq, 3DGS ning amaliy qo'llanilishidagi asosiy cheklov – uning katta hajmli sahnalarga moslashmaganligidir. Katta hajmli sahnalarni (masalan, butun shahar ko'chasi, katta me'moriy majmua yoki aerofotosurat asosidagi hudud) 3DGS yordamida rekonstruksiya qilishda Gauss primitivlari soni bir necha milliondan o'nlab milliongacha oshadi. Bu quyidagi muammolarni keltirib chiqaradi:

· GPU xotirasi (VRAM) yetarli emas – zamonaviy GPUlar odatda 24GB xotiraga ega, bu esa 10 milliondan ortiq Gauss primitivi parametrlarini, gradientlarini va optimallashtirish holatlarini saqlash uchun yetarli emas.

· O'qitish vaqti keskin ortadi – har bir iteratsiyada barcha Gauss primitivlarini qayta ishlash hisoblash resurslarini talab qiladi.

· Render vaqtida ham barcha Gauss primitivlari, hatto kameradan uzoqda joylashganlari ham, proyeksiyalanadi, bu esa real vaqt rejimiga xalaqit beradi.

So'nggi tadqiqotlar ushbu muammolarni hal qilishga qaratilgan. Masalan, CityGS-X katta hajmli sahnalarni rekonstruksiya qilishda geometrik aniqlik va samaradorlikni oshirishga harakat qiladi. OccluGaussian esa sahnalarni bloklarga bo'lib rekonstruksiya qilishda okklyuziya muammolarini hal qiladi. Biroq, ko'pchilik mavjud usullar faqat bitta jihatni (masalan, faqat xotirani yoki faqat vaqtni) optimallashtiradi.

Ushbu tezisda katta hajmli sahnalarni 3DGS asosida real vaqtda vizuallashtirishda xotira va hisoblash resurslarini kompleks optimallashtirish strategiyasi taklif etiladi.

Katta hajmli sahnalarni 3DGS yordamida rekonstruksiya qilishdagi asosiy muammolarni quyidagicha tavsiflash mumkin:

Birinchidan, GPU xotirasi cheklanganligi. 3DGS o‘qitish jarayonida har bir Gauss primitive uchun pozitsiya (3), shkala (3), aylanish (4), opasitlik (1) va rang (3) parametrlarini – jami 14 ta parametrni – saqlashi kerak. Bundan tashqari, gradientlar va optimallashtirish holatlari (masalan, Adam optimizatoridagi momentlar) ham xotirani talab qiladi.

10 million Gauss primitive uchun bu taxminan 3-4 GB xotirani tashkil qiladi, 18 million uchun esa 7-8 GB dan oshadi.

Ikkinchidan, render vaqtida barcha Gauss primitivlarining proyeksiyalanishi. 3DGS render jarayonida har bir Gauss primitive kameraning frustumiga tushadimi-yo‘qmi tekshiriladi va tushsa, proyeksiyalanadi. Katta hajmli sahnada primitivlarning aksariyati kameradan uzoqda yoki frustumdan tashqarida bo‘lishiga qaramay, ularning barchasi tekshiriladi, bu esa hisoblash vaqtini oshiradi.

Uchinchidan, LOD (Level-of-Detail) boshqaruvining yo‘qligi. An’anaviy 3DGS da primitivlar orasida hech qanday ierarxiya mavjud emas. Kameradan uzoqdagi obyektlar uchun ham yaqindagidek ko‘p primitiv ishlatiladi, bu esa resurslarning samarasiz sarflanishiga olib keladi.

To‘rtinchidan, sahnalarni bloklarga bo‘lishda chegara artefaktlari. Katta sahnalarni bloklarga bo‘lib rekonstruksiya qilishda bloklar orasidagi chegaralarda vizual artefaktlar (choklar, rang farqlari) hosil bo‘ladi.

Taklif etilayotgan optimallashtirish strategiyasi uchta asosiy komponentdan iborat.

1. Kameraga masofaga asoslangan Level-of-Detail (LOD)

Birinchi komponent – kameraga masofaga qarab optimal Gauss to‘plamini tanlash imkonini beruvchi ierarxik LOD tizimi. Gauss primitivlari bir necha LOD darajalariga ajratiladi:

· L0 (eng yuqori detal): Barcha Gauss primitivlari – yaqin masofadagi obyektlar uchun.

· L1 (o‘rta detal): Kichik va kamopasitli Gauss primitivlari olib tashlangan holda – o‘rta masofadagi obyektlar uchun.

· L2 (past detal): Faqat katta va yuqori opasitlikli Gauss primitivlari – uzoq masofadagi obyektlar uchun.

Har bir LOD darajasi uchun Gauss primitivlari to‘plami depth-aware 3D smoothing filtri qo‘llash va importance-based pruning orqali yaratiladi. Render vaqtida kamera masofasiga qarab mos LOD darajasi tanlanadi. Bu yondashuv Octree-GS da ham qo‘llanilgan bo‘lib, unda o‘sish-pruning strategiyasi orqali Gauss primitivlari mos LOD darajalariga joylashtiriladi.

LOD darajalarini yaratish uchun quyidagi pruning mezonidan foydalaniladi: $Score(g) = \alpha_g \cdot \sigma_g \cdot \Sigma_{g\_F}$

Bu yerda α_g – opasitlik, σ_g – shkala, Σ_{g_F} – kovariatsiya matritsasining Frobenius normasi. Har bir LOD darajasi uchun score qiymati past bo‘lgan primitivlar olib tashlanadi.

2. Visibility-in-block bo‘linish strategiyasi

Ikkinchi komponent – sahnalarni fazoviy bloklarga bo‘lish va har bir blok uchun faqat ko‘rinadigan hududlarni o‘qitishga jalb qilish strategiyasidir. Sahna uch o‘lchamli gridga bo‘linadi (masalan, 50×50×10 m o‘lchamdagi bloklar). Har bir blok mustaqil ravishda o‘qitiladi, biroq bloklar orasidagi chegaralarda artefaktlarni oldini olish uchun maxsus mexanizm

qo‘llaniladi.

Blok chegaralarida vizual artefaktlarni oldini olish uchun quyidagi yondashuv qo‘llaniladi:

1. Overlap hududlari: Qo‘shni bloklar 10-15% overlapga ega bo‘ladi.

2. Opacity blending: Blok chegarasida Gauss primitivlarining opasitligi blok markazidan chetga chiqqan sari asta-sekin kamayib boradi.

3. Boundary-aware optimization: Qo‘shni bloklardagi Gauss primitivlari o‘rtasida atributlar uzluksizligini ta’minlovchi qo‘shimcha regulyarizatsiya hadi qo‘shiladi.

Har bir blok uchun o‘qitish jarayonida faqat shu blokga tegishli Gauss primitivlari va ularga mos keluvchi kamera ko‘rinishlari ishlatiladi. Bu yondashuv xotira sarfini sezilarli kamaytiradi, chunki har bir vaqtda faqat bitta blokning primitivlari GPU xotirasida saqlanadi.

3. Host offloading mexanizmi

Uchinchi komponent – Gauss primitivlarini host (CPU) xotirasida saqlash va faqat kerakli qismini GPUga yuklash mexanizmidir. Bu yondashuv GS-Scale da qo‘llanilgan bo‘lib, GPU xotira talabini 3.3-5.6 baravarga kamaytirish imkonini beradi.

Host offloading mexanizmi quyidagi optimallashtirishlarni o‘z ichiga oladi:

1. Selective offloading: Faqat geometrik parametrlar (pozitsiya va shkala) CPUda saqlanadi, rang va opasitlik kabi parametrlar esa GPUda qoladi. Bu frustum culling operatsiyalarini CPUda tez bajarish imkonini beradi.

2. Parameter forwarding: CPU optimallashtirishni (masalan, Adam update) GPU hisoblashlari bilan parallel bajarish imkonini beruvchi pipeline mexanizmi.

3. Deferred optimizer update: Gradientlari nolga teng bo‘lgan Gauss primitivlari uchun optimallashtirishni kechiktirish orqali xotiraga kirish sonini kamaytirish.

Render vaqtida esa faqat joriy ko‘rinishga (frustumga) tushadigan Gauss primitivlari hostdan GPUga yuklanadi

Ushbu tezisda katta hajmli sahnalarni 3DGS asosida real vaqtda vizuallashtirishda xotira va hisoblash resurslarini kompleks optimallashtirish strategiyasi taklif etildi.

1. Kameraga masofaga asoslangan ierarxik LOD tizimi – bu primitivlarni muhimlik darajasiga qarab saralash va faqat kerakli darajadagi primitivlarni render qilish imkonini beradi.

2. Visibility-in-block bo‘linish strategiyasi – bu sahnalarni bloklarga bo‘lib, har bir blokni mustaqil o‘qitish va blok chegaralarida opacity blending orqali artefaktlarni bartaraf etish imkonini beradi.

3. Host offloading mexanizmi – bu Gauss primitivlarini host xotirasida saqlash va faqat kerakli qismini GPUga yuklash orqali GPU xotira talabini keskin kamaytiradi.

Adabiyotlar, References, Литературы:

1. Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics, 42(4), 1-14.

2. Kulhanek, J., Rakotosaona, M. J., Manhardt, F., Tsalicoglou, C., Niemeyer, M., Sattler, T., Peng, S., & Tombari, F. (2025). LODGE: Level-of-detail large-scale Gaussian splatting with efficient rendering. NeurIPS 2025.

3. Shen, J., Qian, Y., & Zhan, X. (2025). LOD-GS: Achieving levels of detail using scalable Gaussian soup. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 671-680.

4. GS-Scale (2025). GS-Scale: Unlocking large-scale 3D Gaussian splatting training via host offloading. arXiv preprint
5. Octree-GS (2025). Octree-GS: Towards consistent real-time rendering with LOD-structured 3D Gaussians. IEEE Transactions on Visualization and Computer Graphics.
6. CityGS-X (2025). CityGS-X: A scalable architecture for efficient and geometrically accurate large-scale scene reconstruction. ICCV 2025.
7. Consistency-preserving Gaussian splatting for block-based large-scale scene reconstruction (2025). Elsevier.
8. Chen, G., & Wang, W. (2026). A survey on 3D Gaussian splatting. ACM Computing Surveys, 58(12), 1-39.