Dokumentace způsobu řešení

Testování výkonosti kompresního algoritmu Google Draco

Řešeno v rámci předmětu **KIV/ZSWI**

Řešitelé: Jakub Churavý, Daniel Wébr, Jiří Veselý

Zadavatel: Doc. Ing. Libor Váša, Ph.D.

Akademický rok: 2019/2020

Obsah

Zadání práce

Vytvořte wrapper pro knihovnu Google Draco (dále jen Draco) (https://github.com/google/draco), který umožní testování algoritmů implementovaných v této knihovně pomocí testovacího frameworku. Proveďte nutné úpravy frameworku tak, aby testování probíhalo, pokud možno přímočaře na dodané množině vstupních dat a proveďte proměření výkonnosti.

V rámci předmětu ZSWI byl v akademickém roce 2018/2019 vytvořen nástroj pro testování a evidenci výkonnosti kompresních algoritmů určených pro trojúhelníkové sítě (dále jen testovací nástroj).

Práce s Google Draco

Jedná se o knihovnu pro kompresi a dekomprimaci 3D geometrických sítí a bodových mraků (point clouds). Jejím cílem je zlepšit ukládání a přenos 3D grafiky. Pracuje s formáty dat PLY a OBJ.

Zdrojový kód je napsán v programovacím jazyce C++. Data se ale dají dekompresovat i pomocí knihovny napsané v programovacím jazyce Javascript. Způsob pro práci s projektem je více popsán zde: https://github.com/google/draco/blob/master/BUILDING.md V naši práci jsme pracovali výhradně s kódem psaným v jazyce C++. Draco bohužel v době tvoření práce neměl dostačující dokumentaci a většinu informací o povaze algoritmu jsme zjišťovali až přímo při vývoji.

Při kompresi 3D sítí nabízí draci 2 možné způsoby, a to *sequential ecoding* a *edgebreaker*.

Edgebreaker je defaultní přístup, nabízející efektivnější kompresi, ale za cenu reindexace a možnosti ztráty vrcholů. Nabízí 10 úrovní komprese (1-10), které ovlivňují velikost kompresovaného souboru a distorci souboru po dekompresi.

Sequential ecoding je způsob komprese při které nedochází k žádné ztrátě dat a pořadí vrcholů zůstává zachováno. Tento způsob má jenom jednu úroveň komprese a je méně efektivní.

Způsob řešení

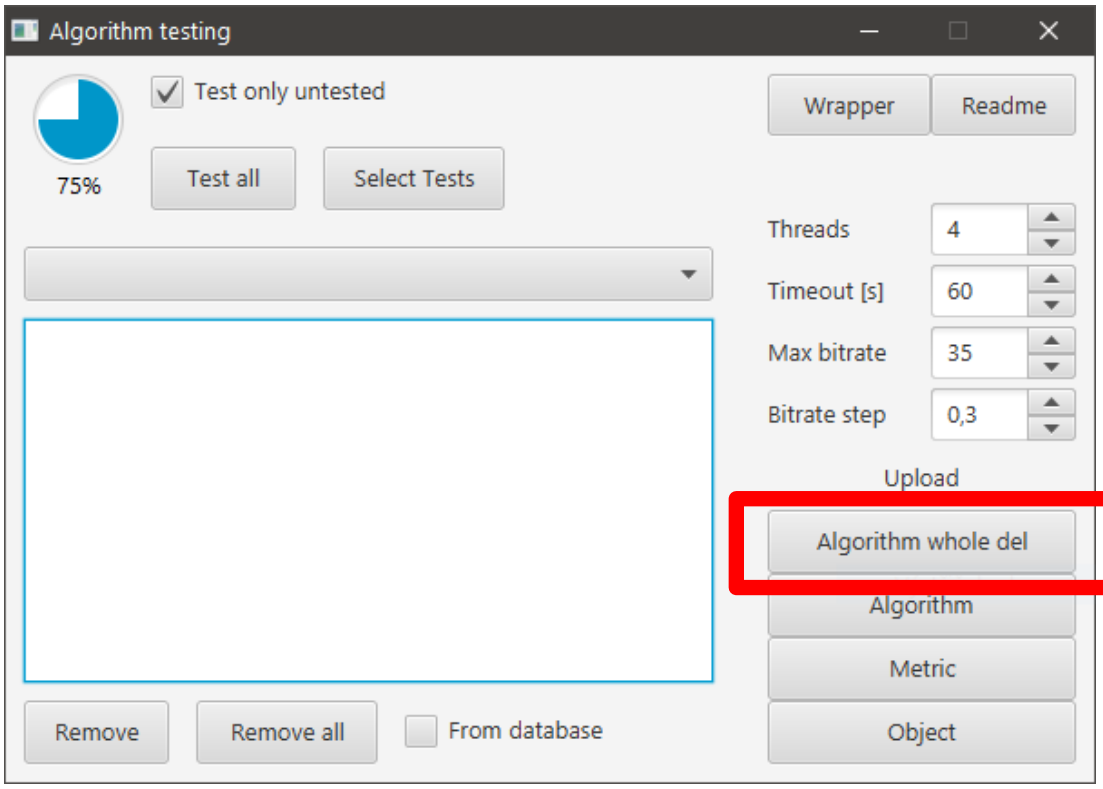
Úkolem práce bylo tedy vytvořit wrapper (interface) mezi Dracem a testovacím nástrojem, který Draco otestuje na připravených metrikách. V již vytvořeném nástroji pro testování dat se nacházely metriky DameCl, Dame_visibility a MSE.

Vytvoření wrapperu bylo nutné, jelikož testovací nástroj přímo spustí kompresní algoritmy z příkazové řádky s daným formátem vstupních parametrů. Nástrojem očekávané soubory .exe jsou:

compress.exe <input.obj> <delta> - vytvoří *input.obj.bin*
decompress.exe <input.obj.bin> - vytvoří *mesh.obj*
reindex.exe <input.obj> <output.obj> - vytvoří reindexed.obj

V rámci práce jsme tedy naprogramovali soubory *draco_decoder.cc a draco_encoder.cc* v jazyce C++, které využívají C++ knihovnu Draco. Soubory jsme vytvořili přímo v projektu Draca. Z těchto zdrojových souborů jsme vytvořili soubory *compress*.exe a *decompress.exe*. Tyto soubory již vyhovovaly specifikacím nástroje.

Dále bylo potřeba upravit testovací nástroj, který testoval algoritmy na kompresích delta 0-1 v rámci desetinných čísel. Na tomto rozmezí Draco úroveň delta nepodporuje, tudíž jsme do testovacího nástroje přidali možnost načtení kompresního algoritmu, který bude testován na delta 1-10.

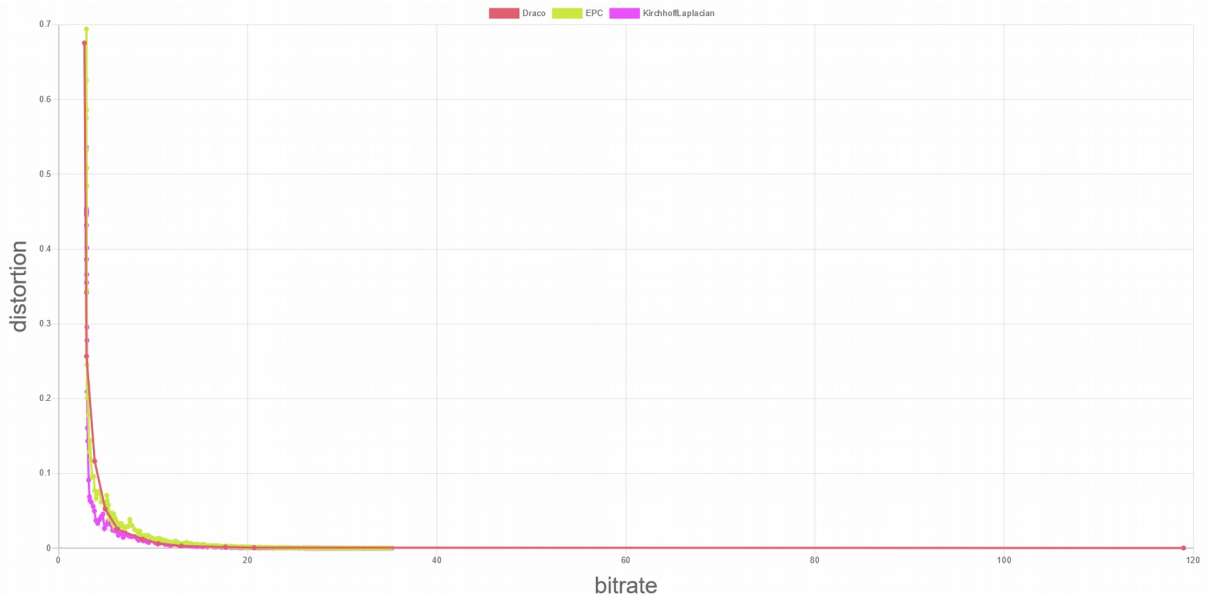


Poté jsme ale narazili na problém, kdy Draco stále nemohl být otestován. Způsoboval to fakt, že výsledné sítě měly přeházené pořadí hran, ale metriky očekávaly pořadí zachované. Pro to, aby byl Draco úspěšně otestován se nabízelo několik řešení:

1. **Nastavit způsob *sequential ecoding*** (viz kapitola Práce s Google Draco). To jsme tedy vyzkoušeli, ale s překvapením jsme zjistili že Draco hrany stále reindexuje. Podařilo se nám přijít na to, že to byl problém pouze novějších verzích Draca. Při použití starší verze (1.2.0) už sequential fungoval, ale stále zbytečně, jelikož existuje jenom jedna úroveň komprese.
2. **Přidat do souboru .obj každému vrcholu parametr rovný jeho indexu.** To by umožnilo po dekompresi souboru vrátit pořadí hran do původního pořadí pomocí vytvořeného programu. Stále by se ale musel řešit problém menšího počtu vrcholu. Navíc přidané atributy by měly negativní vliv na kvalitu komprese a dobu testování.
3. **Zjistit způsob reindexace na nízké úrovni komprese.** Jelikož vrcholy zůstanou pozměněny minimálně bylo by možné pomocí programu dohledat původní pořadí. Takto zjištěné pořadí by se aplikovalo na reindexování sítí s vyššími úrovněmi komprese. Tento způsob také bohužel nemohl být realizován, jelikož způsob reindexace je závislý na úrovni komprese.
4. **Pro testování algoritmu využít metriku nevyžadující stejné pořadí vrcholů**. Tento způsob se nám podařilo realizovat pomocí metriky Metro, kterou jsme přidali do testovacího nástroje.

Pomocí 4. způsob způsobu se nám tedy podařilo, po drobných úpravách testovacího nástroje (speciální výjimka pro Metro, jelikož vracelo dlouhý výpis a nás zajímá pouze výsledek distorce), Draco otestovat.

Výsledky testování



Shrnutí

Úspěšné otestování Draca pomocí testovacího nástroje zahrnovalo:

- Vytvoření wrapperu
  - soubory *draco_encoder.cc a draco_decoder.cc* ze kterých jsme vytvořili soubory .exe (compress.exe a decompress.exe)
- Úprava testovacího nástroje
  - Možnost testování na deltě 1-10
  - Přidání metriky Metro (bylo potřeba pozměnit získávání výsledků, jelikož metro vracelo dlouhý výpis)

Rozdělení GitHub

Složka draco – obsahuje projekt v program VirtualStudio, tento projekt je poskytován samotným Googlem, my jsme vytvořili upravené soubory *draco_encoder.cc a draco_decoder.cc*

Složka zswi-kompresni-algoritmy-master – obsahuje upravený původní testovací software