Konstrukcija slike pomoću optimizacijski algoritama

Barbarić Filip

***Abstract*—**
—U ovome radu je objašnjena metoda konstrukcije slike koja koristi evolucijskoj programiranje, algoritme za uni modalnu optimizaciju bez derivacija i sijenčare. Mi ćemo slučajnu generaciju prilagoditi nekoj slici. Ova metoda je vrijedna razvoja jer imam potencija primjene u grafici a sa time rješava problem generiranja grafičkih objekata koji su za ljude teško modelirali.

I. Uvod

Umjetnost je svugdje oko nas. Danas umjetnici nalaze razne kreativne načine kako bi izrazili svoje vizije. Tako je i umjetna inteligencija postala jedno od sredstava sa kojim se umjetnici izražaju. Predstavit ću popularnu metodu konstrukcije slike pomoću evolucijskoj programiranja.

II. Definicija problema

Za zadanu ulaznu sliku U napravi lažnu sliku I koja je načinjena od niza objekata koji daju sadržaj. Objekti koji daju sadržaj mogu biti poligoni, druge manje slike, različiti matematički opisi sadržaja kao vornojevi dijagrami, slike zaslona 3d modela. Za objekte nam je jedino bitno da se daju opisati uz mali broj varijabli.
Ako bi smo za objekte koristili kvadratić. Onda bi smo morali imati dvije varijable za poziciju, dvije varijable za dužinu i visinu kvadratića, te tri varijable za boju. Što je ukupno 7 varijabli. Ovaj broj se čini jako mali no u našem riješenu će vrlo brzo rasti.
Takoer jedan od problema je preklapanje objekata. U radu sam radio tako da sam stavio manje objekte da budu ispred.

III. Pregled rješenja

Prva stvar koju moramo definirati je funkcija kazne. Najprirodnija funkcija kazne je

$$\mathcal{L} = \sum\sum(I-U)^2.$$

Zbog ograničenja koja ću objasniti kasnije ja sam koristio sljedeću funkciju

$$\mathcal{L} = \sum\sum|I-U|.$$

Naš cilj je funkciju gubitka svesti na što manji broj. To ćemo postići evolucijski algoritmima i algoritmima za uni modalnu optimizaciju. Ja sam odrabrao Vornojeve dijagrame kao objekte prikaza sadražaja.

A. Zapis rješenja

Prvo se moramo dogovoriti kako ćemo naše riješenje zapisati. Svaka jedinka je sastavljena od niza objekata za prikazivanje. Naša populacija je niz takvih jedinki.

B. Evolucijski algoritmima

Evolucijski algoritmima rade na način da generiraju veliki broj rješenja. Ta rješenja će se ocijeniti koliko su dobra, nakon toga ćemo odabirati dva rješenja proporcionalno njihovoj dobroti, tj funkcija selekcije. Kada imamo dva rješenja uzet ćemo pola rješenja od jednog roditelja pola od drugoga, tj križanje. Nakon toga novo dobijeno dijete ćemo mutirati. Ove tri funkcije selekcija, križanje i mutacija ponavljamo dok ne dobijemo novu populaciju.
Ovo je vrlo grubi opis algoritama. No nama je bitno da odredimo te tri funkcije
Za selekciju sam odabrao kTurnir, koji uzima k jedinki i vrača najbolju.
Za križanje sam odabrao algoritam koji uzima pola jednog pola drugog gena.
Mutiranje sam odabrao kao vrlo rijetku promjenu jednog kromosoma za mali iznos.

C. Implementacija

Koristio sam C++, OpenGL i OpenCV.
C++ sam koristio jer sa tim programskim jezikom imam naviše iskustva. Pokazao se kao dobar programski jezik koji nagrauje trud sa efikasnošću no nekad je dosta teško pisati u njemu. Dinamičko čišćenje memorije moramo odraditi sami te sam koristio pametne pokazivače. Bez njih rješenje bi bilo puno neučinkovitije.
OpenGL sam koristio kako bih nacrtao moju sliku pomoću sjenčara. Koristio sam sjenčar fragmenata, vertex, i geometry. A implementaciju Voronojevih dijagrama sam odradio pomoću stožaste metode.
Kod implementacije jako mi je bitna efikasnost. Meni nije bitno vidjeti sliku dok mi ju program optimizira. Meni je jedino bitna funkcija kazane. Tako sam implementirao da program crta sliku u framebuffer. Nakon što sam nacrtao sliku pokrečem drugi sjenčar koji ima teksturu naše ciljne slike. Framebuffer oduzimam od teksture i rješenje šaljem sa grafičke kartice. To sažimanje se odvija preko atomic varijabli. Glavni problem ovdje metode i cijelog mog rada je taj da atomic može biti samo int. Što me jako ograničava jer je int mali. Ali smo dobili veliko ubrzanje. Dobio sam i do 10000 vrednovanih slika po sekundi.
OpenCV sam koristio za obradu slika. Odnosno za

zamučivanja. Razlog zamučivanja je taj što su slike visoko frekventne, odnosno imaju vrlo nagle prelaze. Smatram kako bih dobio bolja rješenja ako zamutim slike. No ja nisam za sada primjerio neku veliku razliku.
Kod sam implementirao na način da je lako izmjenjiv. Odnosno nožem samo implementirati novi sjenčar i dobili bi smo rješenje za taj sječar. Algoritam bi radio isto.

D. Detalji

Kod Voronojevih dijagrama predloženo je rješenje da se nakon većeg broja generacija gen podupla. Doslovno samo kopiramo iste točke na isti gen. To radimo kako bi smo mogili dobi više detalja.
Svrha evoluciskog programiranja je ta da izbjegnemo lokalne optimume. Mi nakon što smo dobili naše zadovoljavjuče riješenje trebamo primjeniti neku od metoda za uni modalno optimizaciju. Derivacije nisu opcija zbog preklapanja objekata. Moramo koristiti metodu kao Hook Jesson, no ona zahtjeva veliki broj ocjenjivanja. Bolja bi metoda bila simpleks po Nelderu i Meadu. On zahtjeva više rješenja ali manje ocjenjivanja. Ta rješenja možemo skupiti tjekom evoluciskog djela u prioriy queue. Pamtimo samo najboljih D+1, gjdje je D ukupni borj parametara. Jedna od ideja koju možemo primjeniti je ta da optimiziramo evoliciski, pokrenemo Nelderu i Meadu, udvostručimo točke radi detalja. I onda to ponovimo neki odreeni borj puta.

IV. Budući rad

Ova metoda bi lako trebala biti primjenjiva na 3d grafiku. Ako uzmemo modele koji su napravljeni od nekog zapisa malog broja elemenata mogli bi smo taj model prilagodit. Tako na primjer algoritam za crtanje drva ili oblaka.
Takoer funkcija gubitka je prirodna no možemo i biti kreativniji. Tako na primjer možemo koristi funkcije prijenosa stila. Nisam siguran kako bi se one mogle ugraditi u moj kod. Nama bi trebala pomoć PyThorca jer radimo sa neuronskim režama. Algorimtam bi bio još sporiji ali mislim kako bi rješenja bili fascinatnija.
Na kraju bih povezao zadnja dva prijedloga i rekao kao bi ovo bio način za primjenu prijenosa sitla na 3d modelima.

V. Rezultati

TABLE I