Zpráva o řešení projektu (BIN 2020)

Autor: xslavk01
Vedoucí zadání: V. Mrázek

Zadání projektu

Seznamte se aproximačními komponenty (8-bitovými sčítačkami a násobičkami), které jsou k dispozici na stránkách výzkumné skupiny (https://github.com/ehw-fit/evoapproxlib, https://ehw.fit.vutbr.cz/evoapproxlib/). Pomocí Kartézského genetického programování navrhněte filtr pro detekci hran v obraze, který bude využívat přesné i některé aproximované komponenty. Navrhněte nejprve filtr s přesnými komponentami a potom se snažte navrhnout filtr, který bude mít příkon (t.j. suma příkonů komponent) 90%, 80%, ... oproti přesnému. Vyhodnoťte také výslednou závislost příkonu a kvality výstupu.

Implementace

Pri implementácií som vychádzal z poskytnutých zdrojových kódov na prvom cvičení (CGP pre binárne obvody). Tento kód som upravil, aby fungoval pre spracovanie obrázkov, čím som ale stratil výhodu paralelného vyhodnocovania, čo spôsobilo značné spomalenie. Ďalej som vychádzal z článku Image Filter Design with Evolvable Hardware od L. Sekaniny a z knihy Cartesian Genetic Programming od J. Millera, kde sa rozoberajú obrazkové filtre.

Evolučný algoritmus sa konfiguruje v súbore `*src/cgp.h*` a preklad sa spúšťa príkazom make. Pri spustení programu je treba parametrom predať vstupný súbor, referenčný súbor a voliteľne názov logovacieho súboru. To aká násobička bude použitá sa vyberá v súbore `*src/chromosome.cpp*` zakomentovaním a odkomentovaním preprocesorových príkazov.

Selekciu robím elitizmom, teda nová generácia sa vytvára z génu s najlepšiou fitness hodnotou. V prípade, že viacero jedincov malo zhodnú fitness hodnotu, tak jeden bol vybratý na nahrade jedincov, čo mali horšiu fitness a ostatný jedinci boli ponechaný. Toto sa stávalo pomerne často, kedže nová generácia by bola nahradená jedným jedincom a mutácie nemusia ovplyvniť výstup. Táto úprava urýchlila prehľadávanie.

Mutácie robím na celej novej populácií s výnimkou najlepšieho jedinca, ktorý je zachovaný.

Ako fitness funkciu som použil MSE medzi pixelmy výstupu a referenčného obrázku. Najlepšia fitness hodnota je teda 0.

Použil som 16 operácií, ktoré mohli byť vykonávané nad vstupom jednotlivých funkčných blokov CGP. Každý funkčný blok má dva vstupy **A** a **B**. V tabuľke sú vypísané použité funkcie.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	Max	Min	Add	Sub B	Sub A	&	|	Mul	Not A	Not B	A&~B	NAND	NOR	0xFF	0x0

Nastavení experimentů

V experimentoch najskôr budem skúmať ako konfigurácia ovplyvňuje chybu a potom vyberiem niekoľko konfigurácií, v ktorých skúsim nahradiť sčítanie a násobenie za aproximačné sčítačky a násobičky, a porovnám výsledky.

Pre každý experiment vykonám 10 behov, s maximálnym počtom iterácí 100 tisíc. Optimálizia sa automaticky ukončí, ak prebehlo viac ako 10 tisíc generácií bez zmeny. Veľkosť populácie som zvolil podľa článku od L. Sekaniny, 16 jedincov. Pri experimentovaní s konfiguráciou budem meniť počty funkčných blokov a L-back parameter.

Ďalej porovnám výsledky evolvovaných filtrov, vzhľadom na referenčný obraz a ako sa zlepšujú počas behu.

Ako referenčné dáta som pužil obrázok ascent, ktorý je verejne dostupný a nachádza sa v knižnici scipy. Na vytvorenie referenčných dát som použil Sobel filter, implementovaný v knižnici scipy.

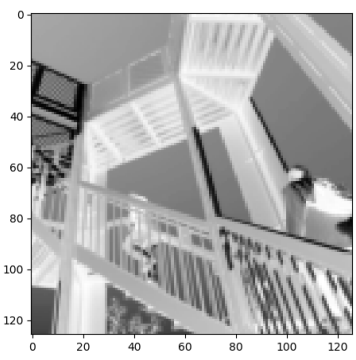
Vyhodnocení experimentů

Testované konfigurácie CGP:

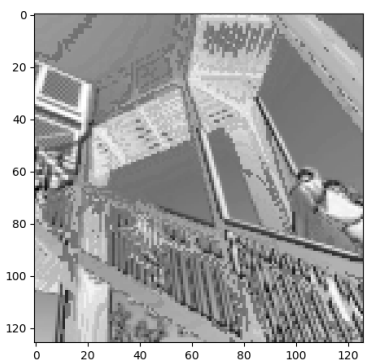
Počet stĺpcov	Počet riadkov	L-Back	Priemerná MSE chyba	Minimálna MSE chyba	Priemerný počet generácií
5	5	1	10652.3020	9716.6787	12818.0
5	5	5	10316.9320	9160.2627	12259.6
7	7	1	9725.8904	9175.3203	10721.3
7	7	7	9700.3073	7999.1069	13457.4
5	7	5	10433.0383	9657.8984	11803.7
7	5	7	9552.3980	8525.9219	11959.6

Z výsledkov prvého pokusu môžeme usúdiť, že konfigurácia s najväčším stavovým priestorom, dosahovala najlepšie výsledky, ale ani jedna nenašla úplne správne riešenie s nulovou chybou. Nájlepší nájdený fenotyp je uložený v súbore `*best_chromosome.chr*`.

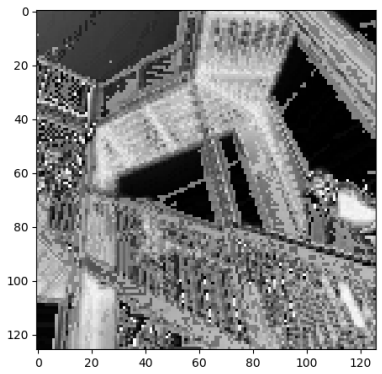
Ukážka evolúcie filtra:



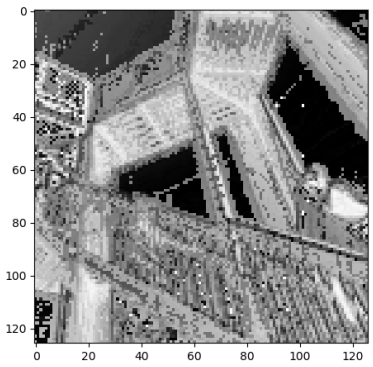
Generation 9



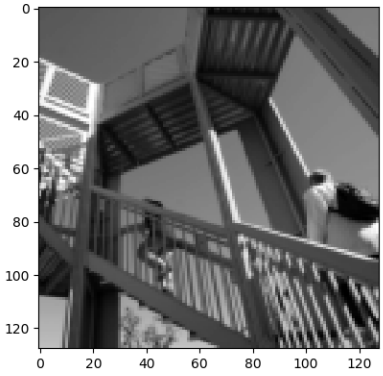
Generation 1196



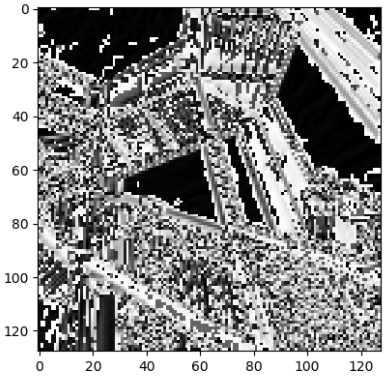
Generation 2308



Najlepší výsledok



Vstup



Referenčný výstup (Sobel filter)

Aproximačná sčítačka a násobička

Pri experimentovaní s aproximačnými komponentami som vyhodnodnocoval nasledujúce kombinácie komponent:

Názov komponenty	MRE	Pow	Minimálna MSE chyba z 10 behov
add8u_0FP	0.000 %	0.033 mW	9878.399414
mul8u_1JFF	0.000 %	0.391 mW	
add8u_5QL	0.40 %	0.024 mW	8574.1328
mul8u_EXZ	0.033 %	0.380 mW	
add8u_5HQ	1.80 %	0.017 mW	8761.2529
mul8u_2AC	1.25 %	0.311 mW	

add8u_099	6.23 %	0.0095 mW	8574.1328
mul8u_185Q	4.16 %	0.206 mW	
add8u_08V	24.87 %	0.0015 mW	8847.0000
mul8u_FTA	13.96 %	0.084 mW	

Všetky použité komponenty som prevzal zo stránok https://ehw.fit.vutbr.cz/evoapproxlib/. V experimente som vždy použil vybranú dvojicu komponentov. Najlepšie fenotypy zo všetkých piatich experimentov s aproximačnými komponentami sú uložené v súboroch `*add8u_<meno>_mul8u_<meno>.chr*`.

Závěr

Nepodarilo sa mi získať filter, ktorý by mal nulovú chybu a to ani pri dlhšom behu bez skorého ukončenia. Toto mohlo byť spôsobené tým, že som nepoužil najvhodnejšiu selekciu, aj keď moja úprava pomohla, myslím si, že populácie nedosiahli dostatočnú variabilitu.

Pri porovnaní funkčnosti filtrov, v rôzných generáciach je vidieť, že sa blíži očakávanému výsledku.

Použitie aproximačných komponent nemalo významný vplyv na chybu, ale keďže ani presné komponenty nedosiahly nulovú chybu, tak sa nedá povedať, že ich použitie je rovnako efektívne.