Cannyho detekce hran

Petr Dvořáček – xdvora0n
Olivér Lelkes – xlelke00
Gabriel Lehocky – xlehoc01

May 7, 2015

1 Zadání

Vytvořte aplikaci realizující Cannyho hranový detektor. Vstupními parametry detektoru jsou: standardní odchylka Gaussova filtru a horní a dolní hranice hystereze při prahování. Implementujte dvě verze filtru - demonstrační, složenou z jednotlivých kroků detekce zvlášť, funkce pro Gaussovo rozmazání, potlačení nemaxim, apod., a optimální, jedna efektivně pracující funkce.

2 Cannyho detekce hran

S detekcí hran se můžeme setkat v mnoha odvětvích, například ve zpracování obrazu, počítačovém vidění, či v robotice. Cannyho detektor hran byl definován v roce 1986 Johnem Cannym na základě vlastností, které musí splnit detekce hran [1]. V práci Johna Cannyho se tvrdí, že optimální detekce hran musí splňovat určité požadavky na kvalitu, přesnost a jednoznačnost.

Kvalitou detekce se rozumí, že musí být nalezeny všechny hrany v obraze. Zároveň nesmí být detekovány hrany, které jimi nejsou. Přesnost detekce znamená, že nalezené hrany musí být tak blízko skutečné hraně v obraze, jak je to jen možné. Detekovaná hrana je jednoznačná, pokud nalezená hrana v obraze je označena jako hrana pouze jednou. Nesmí docházet ke vzniku vícenásobné odezvy na jednu hranu.

John Canny navrhl algoritmus, které se snaží splnit tyto požadavky. Pro dvourozměrný obraz se implementuje jako algoritmus složený ze čtyř částí:

- Eliminace šumu
- Výpočet magnitudy gradientu
- Výpočet směru gradientu a ztenčení hran
- Prahování s hysterezí

První krok je zásadní, neboť potlačení šumu ovlivňuje výsledek dalších kroků algoritmu. K odstranění šumu se používá konvoluce Gaussovým filtrem, jehož kernel bývá běžně reprezentován maticí 5x5. Jádro lze vypočítat dle rovnice:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Konvoluční matice však může být větší pro větší rozmazání zdrojového obrázku. V této práci budeme však používat běžnou velikost Gaussova jádra (tedy jádro o velikosti 5x5). Celočíselné metody implementace, v0 a v2, (viz dále) zanedbaly první část Gaussové funkce (konkrétně část před e), která normalizuje kopeček Gaussovky tak, aby její integrál byl roven jedné. Gaussova funkce se násobila celočíselnou konstantou. Po konvoluci se provedla normalizace na požadovanou hodnotu (bitovým posuvem, či podělením maximální hodnotou v rozmazaném obrazu).

Druhý krok spočívá ve výpočtu magnitudy gradientu hrany. Pro tyto účely se používají nejčastěji Sobelovy operátory. Existují varianty používající i jiné operátory, např. Robertsův, či Prewittův. Zobrazení magnitudy samo o sobě hrany detekuje, avšak nesplňuje Cannyho požadavky na optimální detektor hran. Pro zjištění magnitudy se využívá vztahu

$$M(x,y) = \sqrt{S_H^2 + S_V^2}$$

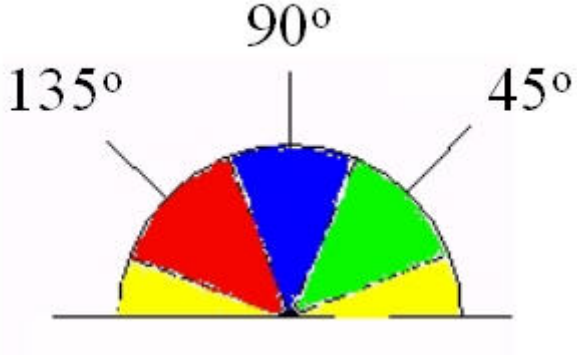

Figure 1: Přibližné úhly hran

1	0	-1	1	2	1
2	0	-2	0	0	0
1	0	-1	-1	-2	-1

Table 1: Sobelovy jádra

kde S_H resp. S_V značí horizontální resp. vertikální složku Sobelova či jiného filtru (metoda v0). Tento vzorec můžeme aproximovat na

$$M(x, y) = |S_H| + |S_V|$$

která je snadnější na výpočet (metody v1, v2).

Cílem dalšího kroku je ztenčit nalezené hrany podle lokálních maxim. Pro každou magnitudu musí být zařazen směr gradientu, a to do čtyř skupin na úhly 0, 45, 90 a 135, viz obrázek 1. Směr gradientu se obvykle počítá dle vztahu

$$\Theta = \tan^{-1}(S_H/S_V)$$

Podle normály tohoto přibližného úhlu se provede ztenčení. Pokud jsou obě sousední magnitudy menší než daná magnituda, jedná se o hranu.

Posledním krokem je dvojité prahování a hystereze. Body obrazu, které mají vyšší magnitudu než zvolený vyšší práh, jsou automaticky řazeny jako hrana. Body obrazu, které mají magnitudu mezi nižším a vyšším prahem, jsou hrany právě tehdy, když jsou napojeny na bod, který byl označen za hranu. Tento krok tedy odstraňuje slabé hrany, které nejsou napojeny na silné.

3 Navržená řešení

Z předchozího textu vyplynulo, že byly navrženy tři způsoby implementace. První z nich (v0) byla vysoce neefektivní. K získání magnitudy byl použit Sobelův filtr (viz tabulku 1), jehož konvoluce s obrázkem vyžadovala násobení a byla spouštěna pro každý operátor zvlášť. Ztenčení hran probíhalo podle přesných normál, které se následně váhovaly. Hystereze se prováděla pomocí rekurze. Tato metoda pracovala na celočíselné aritmetice.

Druhá navržená metoda (v1) využívala jiný výpočet Gaussového kernelu, který byl prováděn podle Gaussovy funkce, jejíž integrál je roven jedné, viz předchozí kapitolu. Tato metoda využívala Prewittův operátor (viz tab. 2). Konvoluce pomocí tohoto operátoru byla prováděna pomocí sčítání a odčítání. Jedním průchodem byla vypočtena vertikální a horizontální složka. K získání normál byla využita funkce $tanh(x)$. Hystereze se prováděla pomocí zásobníku. Tato metoda pracovala na úrovni čísel s plovoucí řádovou čárkou.

1	0	-1	1	1	1
1	0	-1	0	0	0
1	0	-1	-1	-1	-1

Table 2: Prewittovy jádra

	v0	v1	v2	CGP
Obr. 1	5.806666	3.7633330	3.1099990	0.503334
Obr. 2	8.560000	5.7166670	4.4766670	0.733333
Obr. 3	3.086667	1.3866670	1.1966670	0.276667
Obr. 4	8.556666	5.5866670	4.6566660	0.773333
Obr. 5	6.776666	4.4199990	3.5566660	0.623333
AVG	32.786665	20.873333	16.996665	2.19

Table 3: Porovnání navržených metod a evolučního detektoru hran

	Rekurze	Fronta
Obr. 1	1.6913000	1.7666000
Obr. 2	2.4980000	2.6503000
Obr. 3	0.9115000	0.9824000
Obr. 4	2.5469000	2.9317000
Obr. 5	1.9971000	2.1829000
AVG	9.6448	10.5139

Table 4: Porovnání rychlosti hystereze pro verzi v2 s parametry $\sigma = 2$, $T_H = 42$, $T_L = 0$. Byla testována pouze část s hysterezí, nikoli úplná detekce.

Třetí navržená metoda (v2) byla totožná s metodou v1 avšak pracovala na celočíselné úrovni. Pro ztenčení se nevyužívala funkce *tanh*. Pro výpočet přibližných úhlů se braly jednotlivé Prewittovy složky, dle nichž se vypočetl přibližný úhel.

Tyto tři metody byly porovnány na vlastní sadě megapixelových obrázků s parametry $\sigma = 2.0$, $T_H = 0$, nižší práh nebyl definován. Vizte tabulku 1. Z výsledků vyplynulo, že celočíselná aritmetika je rychlejší než aritmetika v číslech s plovoucí řádovou čárkou. Dále se podařilo urychlit řešení oproti původní metodě přibližně dvojnásobně.

Další test se snažil zjistit, zda je rychlejší hystereze dle rekurze, či pomocí zásobníku. Z testu, viz tabulka 4, vyplynulo, že rekurze je rychlejší. Důvod můžeme shledat v paměti cache, kde rekurzivní volání má snadnější přístup. Avšak použití rekurzivní verze nedoporučujeme. Může nám totiž prasknout zásobník.

4 Detekce pomocí evolučně navrženého filtru

Pomocí evolučního návrhu byly vyvinuty různé inovativní řešení, která mohou být lepší než konvenční řešení navržená člověkem. CGP můžeme použít například v návrhu obvodů [2], ve filtraci obrazu, či k detekci hran [3].

4.1 Reprezentace filtru v CGP

Kandidátní filtr bývá nejčastěji reprezentován acyklickým orientovaným grafem. Uzly grafu jsou uspořádány do matice o n_c sloupcích a n_r řádcích. Každý uzel představuje určitou operaci g z konečné množiny operací Γ, viz tabulka 3. Parametr l-back l definuje propojitelnost uzlů mezi jednotlivými sloupci grafu. Pro $l = 1$ je propojitelnost minimální, protože se propojují uzly mezi sousedními. Pro $l = n_c$ je propojitelnost maximální, protože se mohou propojovat libovolné sloupce. Zároveň musí být určen počet primárních vstupů filtru n_i a počet primárních výstupů n_o. Zpravidla bývá $n_i = 9$ a $n_o = 1$, kde vstup filtru označuje devíti okolí.

Zapojení filtru je zakódováno chromozomem konstantní délky. Každý uzel grafu je tvořen trojicí (i_1, i_2, α), kde α představuje kód funkce z Γ, a kde i_1 a i_2 jsou indexy libovolného uzlu předchozích sloupců nebo se jedná o zapojení na primární vstup filtru. Chromozóm dále obsahuje n_o genů, které označují uzly nebo primární vstupy připojené na primární výstup. Příklad zakódování CGP je uveden na obrázku 2.

4.2 Fitness funkce

Jako fitness funkci můžeme zvolit vztah 1, kde $g(x, y)$ je výsledek filtrace pixelu pomocí kandidátního filtru, a kde $t(x, y)$ odpovídá požadované hodnotě daného pixelu. Proměnné x, y odpovídají pozici na

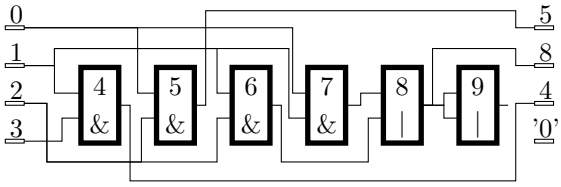

Figure 2: Propojení grafu CGP s parametry: $n_i = n_o = 4$, $n_c = l = 6$, $n_r = 1$, $\Gamma = \{0^{\&}, 1^{|}\}$. Chromozom: $(1,3,0)(0,2,0)(1,2,0)\ (0,1,0)(7,6,1)(8,8,1)(5,8,4,'0')$. Převzato z [2].

daném obrázku, který má C sloupců a R řádků. Tato fitness je přejatá z práce [3].

$$fit = \sum_{x=1}^{R} \sum_{y=1}^{C} g(x,y) - t(x,y) \tag{1}$$

4.3 Prohledávací algoritmus

1. Vytvoří se $(1 + \lambda)$ náhodných filtrů.
2. Každý filtr se ohodnotí fitness funkcí fit.
3. Filtr s nejnižší fitness hodnotou je označen za rodiče. Pokud existuje více jedinců s nejnižší fitness hodnotou, vybere se za rodiče ten jedinec, který jim nebyl v předchozí iteraci (generaci).
4. Z rodiče je vygenerováno λ potomků, na které se aplikuje nanejvýš h validních bodových mutací.
5. Kroky 2–4 se opakují tak dlouho, než je vyčerpán počet generací n_g, či nalezen patřičný filtr.

4.4 Experimentální vyhodnocení evolučních fitlrů

Problém při návrhu evolučních filtrů spočívá v uváznutí evolučního algoritmu v lokálním extrému – můžeme totiž obdržet obrázek vyplněný pouze černou barvou. Proto je potřeba spustit běh CGP několikrát a s omezujícími podmínkami. Pro 100 běhů byly zvoleny následující parametry: $n_c = 10$, $n_r = 5$, $l = 1$, $n_g = 500$, $h = 5$, $\lambda = 4$, $\Gamma = \{255; x; \neg x; x \vee y; x \wedge y; \neg(x \vee y); \neg(x \wedge y); x \oplus y; x/2; x/4; (x * 16) \wedge (y/16); x + y; x +^s y; (x + y)/2; min(x,y); max(x,y)\}$ [3]. Jako vstupní obrázek byl vybrán standardní obrázek kameramana. Referenční obrázek měl detekované hrany dle Cannyho detekce $\sigma = 2, T_H = 40, T_L = 20$.

Na obrázku 3 vidíme nejlépe vyvinuté filtry. Chceme-li dosáhnout lepších výsledků, musíme spustit algoritmus s velkým počtem generací (řádově miliony). S počtem generací a velikostí obrázků přichází problém, který se nachází v době ohodnocení kandidátního řešení. Jeden běh trvá řádově hodiny. Tento problém lze eliminovat různými akceleracemi algoritmu (nejlépe FPGA implementace algoritmu).

Jako akceleraci jsme využili techniku citace na již vyvinutý evoluční detektor hran [3]. V této práci však autor neprovedl rychlostní porovnání a porovnání detektoru na jiných obrázcích. V tabulce 1 můžeme pozorovat rychlost detekce evolučního detektoru. Z tabulky plyne, že lze zrychlit detekci 19x oproti původnímu řešení. Při pohledu na evoluční detektory vidíme, že detekce hran selhává na jiné typy obrázků, než pro které byly vyvinuty. Viz obrázek 4. Další nevýhodou je rychlost nalezení evolučního filtru.

5 Závěr

Byly prezentovány různé metody implementace algoritmu Cannyho detektrou hran. Nejrychlejší se jeví celočíselná aritmetika, avšak implementace využívající čísla s plovoucí řádovou čárkou dávala lépe detekované hrany. Celkově se nám podařilo urychlit algoritmus dvojnásobně.

Byla ukázána metoda pomocí evolučního algoritmu. Hlavní výhoda evolučního detektoru se nachází v rychlosti detekce, která je přibližně 19x rychlejší než původní řešení. Nevýhodou se jeví časová složitost k nalezení řešení. Další nevýhodou je v použití na jiné typy obrázků než, pro které byl detektor vyvinut.

Pokračování práce vidíme v eliminaci problémů evolučních detektorů hran a hledání dalších akceleračních metod pro Cannyho detekci hran.

Figure 3: Nalezené evoluční detektory hran

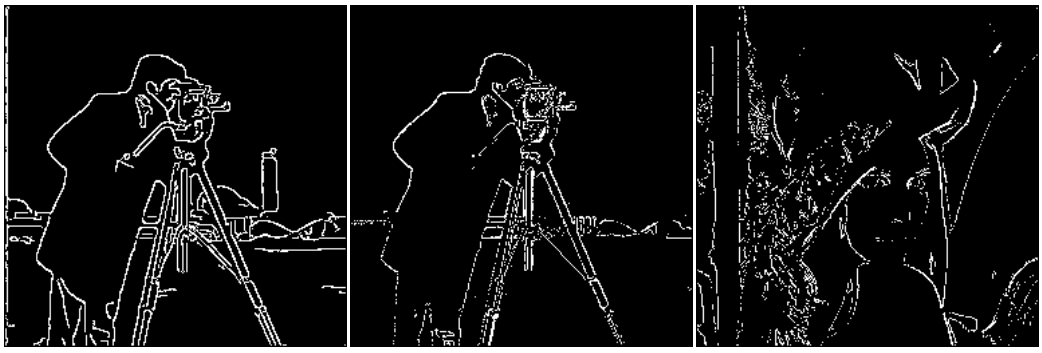

Figure 4: Evoluční detekce hran. Zprava detekce podle Cannyho, vyvinutý evoluční detektor, použití evolučního detektoru na obrázku z časopisu Playboy.

Literatura

.

[1] John Canny: *A computational approach to edge detection*, Pattern Analysis and Machine Intellegence, IEEE Transactions on, 1986

[2] Petr Dvořáček: *Evoluční návrh pro aproximaci obvodů*, Excel@FIT, Brno, 2015

[3] Radek Hrbáček: *Koevoluční algoritmus v FPGA*, diplomová práce, Brno, FIT VUT v Brně, 2013

Rozdělení práce

- Petr Dvořáček – původní implementace, evoluční filtry
- Olivér Lelkes – optimalizace algoritmu
- Gábor Lehocky – implementace a porovnání hystereze, výběr testovací sady

Přiložené soubory

- `./v0_edge_detector/` – původní implementace
- `./v1_canny_edgedetector/` – optimalizovaná verze pracující s čísly s plovoucí řádovou čárkou
- `./v2_canny_edgedetector/` – optimalizovaná verze pracující na celočíselné úrovni
- `./v3_evolved_detector/` – evoluční detekce
- `./cgp_version/` – algoritmus CGP
- `./cgp_output/` – výsledek nalezených evolučních filtrů
- `./test_pics/` – testovací sada
- `./tex/` – zdrojové soubory LaTeX
- `./` – porovnání výpočtu magnitudy