Programatorska Dokumentacia

Jakub Kopal

February 24, 2018

1 Uvod

1.1 Spustenie

Solution obsahuje 2 projekty. *SegmentationLibrary* je staticka kniznica ktorej hlavnej metode *Process* sa predaju vstupne argumenty. Cela segmentacia je implementovana v tejto kniznici. *Segmentation* je konzolova aplikacia ktora len preda vstupne parametre danej kniznici. Pri spusteni *Segmentation.exe* cez prikazovy riadok je potrebne zadat 4 alebo 5 parametrov.

1. cestu k vstupnemu obrazku
2. cestu k vystupnemu obrazku
3. Blur konstantu (idealne 0.5 - 1.5)
4. Treshold konstantu (idealne 100 - 200)
5. min velkost segmentu (optional)

1.2 Motivacia

Program je optimalizovany na casovu a pamatovu zlozitost. V oboch pripadoch sa jedna o O(n), kde n je pocet pixelov. Dalej bolo cielom znizit multiplikantne konstanty, najma pri udrziavani grafu v pamati. Pri optimalizivani casovej zlozitosti sa jedna napr. o potrebu nahradit delenie shiftovanim.

2 Input/Output

Vstup je obrazok v jpg formate. Vystupny obrazok musi byt takztiez v jpg formate. Pre co najrychlepsie nacitanie, ulozenie a pracu s obrazkom sa pouziva framework popisany tu. Jeho konkretna implementacia sa nachadza v ImageHandler.h a ImageHandler.cpp. Pri vytvarani ImageHandler sa musia zadat cesty k obrazkom. Treti argument sluzi ako informacia o uspechu/neuspechu inicializacie Imagehandleru. Metoda Create sluzi na naplnenie argumentu vstupnym obrazkom, metoda Save naopak na ulozenie daneho argumentu. Create

a Save maju argument typu Image, ktory obsahuje vysku, sirku, pocet pixelov a pole pixelov.

3 Struktura programu

Na segmentovanie sluzi objekt **Segmentator**, ktory sa vyuziva cez metodu Process. Hlavna metoda Process postupuje nasledovne :

- Nacita parametre z prikazoveho priadku (ReadParams)
- Nacita vstupny obrazok (ImageHandler::Create)
- Na dany obrazok aplikuje GaussianBlur (GaussianBlur::Process)
- Z daneho zblurovaneho obrazku vytvori graf (Graph)
- Vytvoreny graf segmentuje (Segment)
- Vysegmentovany graf prevedie na obrazok (GraphToImage)
- Ulozi vysledny obrazok (ImageHandler::Save)

4 GaussianBlur

Pred tym nez z daneho obrazku vytvorime graf, je potrebne na obrazok najskor aplikovat blur. Na tento ucel bol vytovreny objekt **GaussianBlur**. Ten bol implementovny casovo najefektivnejsim sposobom, a to v linearnom case. Konkretna implementacia bola odvodena z tohto clanku.

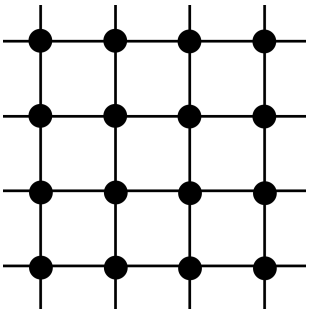
5 Graf

5.1 Reprezentacia

Graf je mriezkoviteho typu. Kazdy pixel obrazku je vrchol ktory je spojeny ohodnotenou hranou s jeho 4 susedmi. Vrchol je reprezentovany objektom typu **Node** ktory si pamata vsetky potrebne informacie(predka, vysku, treshold...). Kvoli uspore pamate su hrany reprentovane len polom intov, kde vrcholi ktore hrana spaja sa zistuju z indexu hrany v poli.

5.2 Inicializacia

Kazdy vrchol je sam sebe predkom. Hodnoty *InternalDifference, Count, Height* su vsetky inicializovane na 1. Preco su tieto hodnoty dolezite bude uvedene neskor v casti 5.3. Hodnoty hran su rozdieli pixelov v *GrayScale* ktore spajaju.



5.3 Segmentacia

Segmentaciu inicializovaneho grafu prevadza metoda *Segment* v nasledujucom poradi
1. Sort hran Sortovanie hran vykonava metoda *EdgeSort* CountingSortom vdaka obmedzenemu poctu hodnot hran (0 - 255). Kvoli potrebe zachovavat poradie(z poradia sa zistuju vrcholy ktore hrana spaja) sa s povodnym polom nehybe, ale namiesto toho sa vysledok ulozi do noveho pola pointrov na povodneho hodnoty. Index n-tej hrany sa potom zisti ako rozdiel adresy v pamati prvej a n-tej hrany.
2. Mergovanie Hrany sa prechadzaju v zotriedenom poradi. Z kazdej hrany sa vezmu vrcholy ktore spaja. Pre oba vrcholy sa najdu korene stromov v ktorych lezia, kazdy vrchol ma odkaz na predka (na zaciatku je kazdy vrchol koren). Ak je *Treshold* oboch korenov vacsi ako hodnota danej hrany, a korene su navzajom rozne, dochadza k mergovaniu stromov. Koren "nizsieho" stromu sa pripoji ako syn ku korenu "vyssieho" stromu. Samotne mergovanie a aktualizaciu potrebnych hodnot vykovana funkcia *Merge*.

5.4 GraphToImage

Vysledne vysegmentovany graf je treba niak vyzualne reprezentovat. O to sa stara funkcia *GraphToImage*. Ta funguje nasledujucim sposobom

1. Zisti pocet korenov grafu
2. Kazdemu korenu priradi "nahodnu" farbu
3. Kazdemu vrcholu grafu najde koren stromu v ktorom vrchol lezi a vrcholu priradi farbu jeho korena.