Semestrální práce z KIV/PC

Přebarvování souvislých oblastí ve snímku

Jiří Velek
A20B0269P
jvelek@students.zcu.cz

26. 10. 2021

Obsah

1 Zadání

Naprogramujte v ANSI C přenositelnou konzolovou aplikaci, která provede v binárním digitálním obrázku (tj. obsahuje jen černé a bílé body) obarvení souvislých oblastí pomocí níže uvedeného algoritmu Connected-Component Labeling z oboru počítačového vidění. Vaším úkolem je tedy implementace tohoto algoritmu a funkcí rozhraní (tj. načítání a ukládání obrázku, apod.). Program se bude spoučtět příkazem

ccl.exe ⟨ input-file[.pgm] ⟩ ⟨ output-file ⟩

Symbol ⟨ input-file ⟩ zastupuje jméno vstupního souboru s binárním obrázkem ve formátu Portable Gray Map, přípona souboru nemusí být uvedena; pokud uvedena není, předpokládejte, že má soubor příponu .pgm. Symbol ⟨ output-file ⟩ zastupuje jméno výstupního souboru s obarveným obrázkem, který vytvoří vaše aplikace. Program tedy může být během testování spuštěn například takto:

```
...\>ccl.exe e:\images\img-inp-01.pgm e:\images\img-res-01.pgm
```

Úkolem vašeho programu tedy je vytvořit výsledný soubor s obarveným obrázkem v uvedeném umístění a s uvedeným jménem. Vstupní i výstupní obrázek bude uložen v souboru ve formátu PGM, který je popsán níže. Obarvení proveďte podle níže uvedeného algoritmu.

Testujte, zda je vstupní obraz skutečně černobílý. Musí obsahovat pouze pixely s hodnotou 0x00 a 0xFF. Pokud tomu tak není, vypište krátké chybové hlášení (anglicky) a oznamte chybu operačnímu prostředí pomocí nenulového návratového kódu.

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, makefile pro Windows i Linux(pro překlad v Linuxu připravte soubor pojmenovaný makefile a pro Windows makefile.win) a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

2 Analýza úlohy

Úloha je rozdělena do tří částí: validace a načítání PGM souboru, přebarvení souvislých komponent, a nakonec zápis do nového PGM souboru.

2.1 Načítaní PGM souboru

PGM soubor je definován následovně:

1. řetězec “P5“
2. bílý znak
3. šířka
4. bílý znak
5. výška
6. bílý znak
7. maximální hodnota šedi, které mohou jednotlivé pixely nabývat
8. bílý znak
9. jednotlivé pixely (šířka $*$ výška pixelů)

V našem případě je třeba kontrolovat, zda každý pixel obsahuje hodnotu 0 nebo 255 a pokud ne, program skončí chybovou hláškou. Zároveň je třeba kontrolovat, zda soubor opravdu obsahuje správný počet pixelů a všechny kontrolní parametry (šířku, výšku, maximální hodnotu šedi). Po dokončení načítání si program uchová pouze výšku obrázku, šířku obrázku a sekvenci jednotlivých pixelů.

2.2 Přebarvování souvislých komponent

Problém přebarvování souvislých komponent lze řešit dvěma různými způsoby:

1. Pomocí algoritmu DFS
2. Pomocí algoritmu CCL

2.2.1 Řešení pomocí algoritmu Depth-first search

Algoritmus depth-first search je algoritmus na prohledávání grafu do hloubky. Samotné řešení našeho problému je rozděleno na dvě funkce: 'find_components' a 'dfs'.

'find_components'

Funkce vytvoří pole 'labels' velikosti $šířka * výška$, které značí, zda už se algoritmus na dané pozici vyskytl, a zároveň si uchová hodnotu oblasti, ve které se daný pixel nachází. Potom pro každý pixel, který ještě není v poli, zavolá dfs, a inkrementuje label, který přiřazuje pixelům.

'dfs'

Pokud předaný pixel neobsahuje label, neobsahuje barvu pozadí (černou), a zároveň jeho souřadnice nejou mimo hranice velikosti obrázku, pak funkce nastaví label předanému pixelu a pro všechny jeho sousedy rekurzivně zavolá sama sebe.

Komplexita algoritmu

První cyklus projde přes všechny pixely - $\mathcal{O}(šířka * výška)$, algoritmus dfs má komplexitu $\mathcal{O}(V + E)$, kde V je počet pixelů, a E počet hran (každý pixel má 8 sousedů), tedy $E = 4 * šířka * výška - 3 * šířka - 3 * výška + 2$, celková komplexita algoritmu je tedy: $\mathcal{O}(šířka * výška)$.

2.2.2 Řešení pomocí algoritmu Connected component labeling

Algoritmus Connected component labeling je dvouprůchodový algoritmus. Při prvním průchodu spojí všechny sousedící vrcholy se stejnou barvou do jedné množiny a při druhém průchodu tyto množiny očísluje a přidělí barvu jednotlivým pixelům. Algoritmus má komplexitu $\mathcal{O}(šířka * výška)$, ale násobící konstanta je nižší, než u DFS, tedy algoritmus CCL bude o trochu rychlejší.

3 Implementace

U konkrétní implementace bylo zvoleno řešení algoritmem Connected component labeling. K realizaci tohoto algoritmu je potřeba vyrobit datovou strukturu reprezentující množinu. Množina je v programu realizována jako algoritmus union-find reprezentovaný polem.

3.1 Union-find

Union-find je algoritmus pro reprezentaci množin a k jejich jednoduchému slučování. Ke sloučení dvou množin A, B stačí pouze najít "rodičovský vrchol množiny A" a přepsat jeho rodiče na libovolný prvek z množiny B.

Union-find je v programu implementován v souborech *set.h*, *set.c*.

3.1.1 Operace find

Operace find najde "rodičovský vrchol množiny", jako parametr přebírá jakýkoliv prvek množiny. V cyklu prochází pole tak, že: $i = a_i$, dokud platí: $a_i \neq a_{a_i}$, potom vrátí index i.

Implementace v jazyce C

```
1 uint8_t find(uint8_t set[], uint8_t element) {
2    while(set[element] != element) {
3       element = set[element];
4    }
5
6    return element;
7 }
```

Zdrojový kód 3.1: Operace find

3.1.2 Operace union

Operace union přebírá dva prvky z různých množin a spojí je do jedné. Nejprve zavolá find na obou prvcích a potom jednomu z nalezených rodičů nastaví rodiče na druhého z nalezených.

Implementace v jazyce C

```
1 /* funkce je pojmenovana onion, nebot v C je union rezervovane
     klicove slovo */
2 void onion(uint8_t set[], uint8_t elementA, uint8_t elementB) {
3    uint8_t a = find(set, elementA);
4    uint8_t b = find(set, elementB);
5
6    set[a] = b;
7 }
```

Zdrojový kód 3.2: Operace union

3.2 CCL

4 Uživatelská příručka

5 Závěr