ZADÁNÍ SEMESTRÁLNÍ PRÁCE
PŘEBARVOVÁNÍ SOUVISLÝCH OBLASTÍ VE SNÍMKU

Zadání

Naprogramujte v ANSI C přenositelnou[1] **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 spouštět příkazem

`ccl.exe`[2] ⟨`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[3] 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[4].

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.

Algoritmus

Přebarvování souvislých oblastí (*Connected-Component Labeling*, CCL) je dvouprůchodový algoritmus z oblasti počítačového vidění. Jeho vstupem je binární obrázek, tj. takový, který obsahuje jen bílé a černé pixely. Bílé pixely představují jednotlivé objekty, které se tento algoritmus pokouší izolovat a označit různými barvami, resp. hodnotami intenzity šedé barvy (tedy tzv. labely, česky štítky, značkami). Černé pixely pak představují pozadí.

Algoritmus CCL pracuje tak, jak je uvedeno v následujícím popisu:

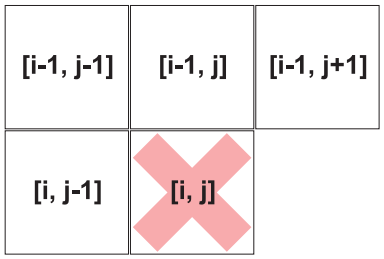
[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10) a s běžnými distribucemi Linuxu (např. Ubuntu, Mint, OpenSUSE, Debian, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 10 Buster s jádrem verze 4.19.0-11-amd64 a s překladačem gcc 8.3.0.

[2] Přípona `.exe` je povinná i při sestavení v Linuxu, zejm. při automatické kontrole validačním systémem.

[3] Nejde o **obarvení** v pravém smyslu slova, protože použitý formát PGM je šedotónový. Obarvením se myslí nahrazení bílé barvy v souvislých oddělených oblastech různými tóny šedé.

[4] Stejně tak čiňte i v případě jiných chyb, např. nesprávném formátu vstupního souboru a podobně.

1. průchod: Procházejte obrázek po řádcích. Každému nenulovému pixelu (tj. pixelu představujícímu objekt) na souřadnicích $[i, j]$ přiřaďte hodnotu podle hodnoty jeho sousedních pixelů, pokud nenulové sousední pixely existují (poloha sousedů je dána maskou na obrázku č. 1). Všechny sousední pixely dané maskou již byly obarveny v předchozích krocích (to je zajištěno tvarem masky).
- Jsou-li všechny sousední pixely součástí pozadí (mají hodnotu `0x00`), přiřaďte pixelu $[i, j]$ dosud nepřidělenou hodnotu – novou barvu.
- Má-li právě jeden ze sousedních pixelů nenulovou hodnotu, přiřaďte obarvovanému pixelu hodnotu nenulového sousedního pixelu.
- Je-li více sousedních pixelů nenulových, přiřaďte obarvovanému pixelu hodnotu kteréhokoli nenulového pixelu ze zkoumaného okolí. Pokud byly hodnoty pixelů v masce různé (došlo k tzv. *kolizi barev*), zaznamenejte ekvivalenci dvojic hodnot do zvláštní datové struktury – tabulky ekvivalence barev.



Obr. 1.: Maska pro přebarvování

2. průchod: Po průchodu celého obrázku jsou všechny pixely náležející oblastem (objektům) obarveny, některé oblasti jsou však obarveny více barvami (díky kolizím). Proto projděte znovu celý obraz po řádcích a podle informací o ekvivalenci barev (z tabulky ekvivalence barev získané v průběhu 1. průchodu) přebarvěte pixely těchto oblastí. Z množiny kolizních barev je možné nějak vybrat jednu (první, poslední, náhodně) nebo opět postupovat při přiřazování barev „od začátku“ a jako novou barvu použít index množiny (nesmí být samozřejmě nulový, protože barva `0x00` představuje pozadí).

Po tomto kroku odpovídá každé oblasti označení (obarvení) jedinou, v jiné oblasti se nevyskytující hodnotou (barvou).

Obrázek č. 2 představuje binární (černobílý) obrázek na vstupu algoritmu. Černé oblasti představují pozadí, bílé oblasti (získané tzv. *prahováním*) představují objekty. Na obrázku č. 3 vidíte výsledek činnosti algoritmu – každý objekt je obarven unikátní barvou. Lze tedy např. zjistit počet objektů ve scéně, zkoumat jejich tvar, apod.

Obr. 2: Vstupní binární obrázek

Obr. 3: Výstupní obrázek s obarvenými oblastmi

Obarvením se zde – v případě šedotónových obrázků – pochopitelně nemyslí změna barvy pixelů v reprezentaci např. RGB (jak již bylo naznačeno), ale označení každé souvislé oblasti jinou úrovní šedi, takže je pak každý „obarvený“ objekt v obrázku snadno identifikovatelný touto svojí „barvou“.

Vstupní a výstupní formát

Na vstupu i výstupu Vašeho programu budiž obrázek v **souboru ve formátu PGM**[5]. Formát tohoto souboru je následující:

`P5`LF	znaky 'P' a '5' a bílý znak* (může být mezera, CR, LF, TAB, VT, FF, ...)
`1024 768`LF	počet sloupců, bílý znak, počet řádek (zapsaný řetězcem znaků), bílý znak
`255`LF	index nejvyšší hodnoty šedé, resp. úrovně jasu (řetězcem znaků), bílý znak
`ČČČuu$$AAAA...`	byty (tj. v jazyce C datový typ `unsigned char`) představující hodnoty jednotlivých pixelů

*) bílým znakem se ve formátu PGM rozumí jakýkoliv znak, pro který vrací funkce ze standardní knihovny překladače jazyka C `int isspace(int c)` nenulovou hodnotu.

Každý pixel je v souboru uložen jako jeden byte (hodnota pixelu je tedy vlastně úroveň šedé). Byte s hodnotou `0x00` znamená úplně černý pixel, zatímco byte s hodnotou `0xFF` znamená úplně bílý pixel.

[5] Formát PGM lze prohlížet a vytvářet např. volně dostupným programem XnView.

Užitečné techniky a odkazy

Uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy víceméně standardní, lze k nim nalézt velké množství dokumentace:

1. Connected-Component Labeling – popis techniky na Wikipedii, `https://en.wikipedia.org/wiki/Connected-component_labeling`
2. Spojový seznam
3. Techniky implementace množin
4. Formát PGM – `http://netpbm.sourceforge.net/doc/pgm.html`

Řešení úlohy je zcela ve vaší kompetenci – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli.