FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

KPG

FILTRY

Martin Hamet 5. května 2016

Obsah

1 Zadání

Nalezené anebo vámi vymyšlené návody na filtry naprogramujte (Program musí být schopen načíst ze souboru obraz v rastrovém formátu a zase jej uložit, zobrazit původní a změněný obraz s možností návratu o 1 akci. Odevzdáváte jako obvykle zdrojový text, EXE a dokumentaci.), za každou zajímavou techniku získáte max. 5 bodů podle obtížnosti, max.17 bodů.

`Splněno:`

- Načítání a ukládání obrázku v rastrovém formátu.
- Možnost návratu o jednu akci.
- Možnost návratu na původní obrázek.
- Implementace filtrů:
  - square blur maticí 3x3
  - square blur maticí 5x5
  - gaussian blur maticí 3x3
  - gaussian blur matici 5x5
  - motion blur maticí 5x5
  - edge (detekce hran) maticí 3x3
  - edge (detekce hran) maticí 5x5
  - embos maticí 3x3
  - "MyExposure"(experimentální filtr) maticí 3x3

2 Analýza úlohy

Pro danou úlohu budeme uvažovat jednoprůchodové filtry. Tedy po získání bitmapy zadaného obrázku, procházíme bitmapou a upravujeme hodnoty(RGB) aktuálního pixelu. Aby nedošlo k ovlivnění filtru směrem průchodu je třeba číst z kopie bitmapy daného obrázku, nebo zapisovat do nové bitmapy. Lze implementovat jednoduché filtry např. pro prohazování nebo potlačování barevných kanálů. Tyto jednoduché filtry mi ale nepřijdou zajímavé a proto se dále budu věnovat pouze konvolučním filtrům.

2.1 Konvoluční filtr

Základní myšlenka konvolučního filtru je určení hodnoty pro každý pixel (při průchodu bitmapou). Tuto hodnotu získáme kombinací okolních pixelů. Tuto kombinaci určí konvoluční jádro.

$$K = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Obrázek 1: Konvoluční jádro

Konvoluční jádro je zpravidla reprezentováno maticí lichého řádu. Konvoluční matice udává váhu jednotlivých okolních pixelů. Tedy jak významně se projeví daný pixel.

Nastavíme konvoluční jádro v bitmapě obrázku tak aby hledaný pixel byl uprostřed jádra a všechny hodnoty pixelů vynásobíme hodnotou na jejich odpovídající pozici v konvolučním jádru a sečteme. Výsledné hodnoty jsou hodnoty našeho hledaného pixelu.

Například podle jádra `K` (viz. Obr.1) hledaná hodnota pixelu (ve středu jádra) vznikne jako aritmetický průměr všech jeho okolních pixelů (průměrem okolních pixelů je myšleno průměr jejich jednotlivých barevných složek RGB). Pro snazší zápis konvolučních jader bude lepší když zavedeme váhu kterou vydělíme výslednou získanou barvu. V předchozím případě by jsme mohli nahradit všechny hodnoty jedničkou a váha by byla 9.

2.2 Square Blur

Tento konvoluční filtr má konvoluční matici K podle Obr. 1. Je to tedy aritmetický průměr okolních pixelů. Pro porovnání bude implementován filtr s konvoluční maticí 3x3 a 5x5.

2.3 Gaussian Blur

Tento filtr pracuje stejně jako `Suare Blur` s tím rozdílem že se již nejedná o aritmetický průměr, ale průměr okolních pixelů s váhou danou gaussovským rozložením podle vzdálenosti od hledaného pixelu. Ukázka konvolučního jádra viz Obr. 2 by měla váhu 159 (váhou je třeba výsledné hodnoty vydělit).

$$\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Obrázek 2: Konvoluční jádro s gaussovým rozložením váhy

2.4 Motion Blur

Filtr simuluje rozmazaní pouze v nějakém směru například pro rozmazání ve vodorovném směru by jádro s váhou 3 mohlo vypadat takto:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

2.5 Edge a Embos

Detekce hran využívá možnosti zadání záporných vah do jádra. Záporná hodnota v jádru znamená že se ve skutečnosti odečtou od hledaného pixelu hodnoty pixelů se zápornou vahou (neboli přičtou se jejich negativní hodnoty). Pomocí záporných hodnot můžeme docílit zvýraznění hran. Jinak řečeno místa kde jsou si pixely podobné se jejich hodnoty navzájem odečtou a tím vzniká černá barva a naopak.

Konvoluční jádro podle Obr. 3 bude detekovat vodorovné hrany v obrázku. Všimněme si že je zde stejná suma záporných i kladných vah. Celkovou váhu kterou by jsme dělili výsledné hodnoty zde neuvažujeme, lze ji ovšem volit pro dolazení efektu. Dělením většími hodnotami ztmavuje obrázek. Dále zde vznikají problémy přetočení hodnot (záporná hodnota některého z kanálu RGB atp.) Je velmi vhodné omezit výsledné hodnoty na maximum a minimum. Bez ošetření obvykle vzniká značný šum v obrázku.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Obrázek 3: Konvoluční jádro pro detekci hran

Embos funguje stejně jako detekce hran s tím rozdílem že k výsledným hodnotám přičteme konstantu tak aby jsme se posunuli z černé barvy na šedé. Experimentálně vhodná hodnota je 128.

2.6 MyExposure

Tento filtr pracuje jako detekce hran s tím rozdílem že nedovolujeme záporné hodnoty. Tedy sčítáme hodnoty kanálů všech pixelů v matici označených jedničkou. Hodnoty kanálů jsou pomocí podmínek omezeny aby se nezvyšovali nad standardní hodnoty na intervalu `<0,1>` stejně jako u detekce hran. Výsledný efekt bývá velmi výrazné zesvětlení již světlých ploch. Je možné toto zesvětlení omezit zvednutím celkové váhy (tj. vydělení příspěvku hodnot jednotlivých pixelů zvolenou konstantou).

3 Uživatelská příručka

Po spuštění programu je třeba načíst obrázek pomocí tlačítka `Load Image`. Po úspěšném načtení obrázku se zpřístupní všechny ostatní volby.

Jednotlivými tlačítky lze používat přednastavené filtry na načtený obrázek. Filtry se skládají za sebe a je možné vrátit poslední akci zpět pomocí tlačítka `Back`. Tlačítkem `Save` je možné uložit aktuální obrázek pomocí ukládacího dialogu.

4 Závěr

Program demonstruje základní efekty pomocí konvoluční matice nad bitmapou obrázku. Všechny typy rozmazání jsou nejznatelnější na obrázcích s malým rozlišením. Při jednom aplikování např. `Gaussian blur` na obrázek s vysokým rozlišením nemusí být efekt znatelný. Podobně rozdíly mezi `Square blur` a `Gaussian blur` v různých rozměrech.

V průběhu implementace jsem experimentoval s detekcí hran v určitém směru v závislosti rozložení hodnot v konvoluční matici, ale ve výsledku jsem ponechal pouze detekci horizontálních hran, které je vhodnější pro např. pro krajinu.