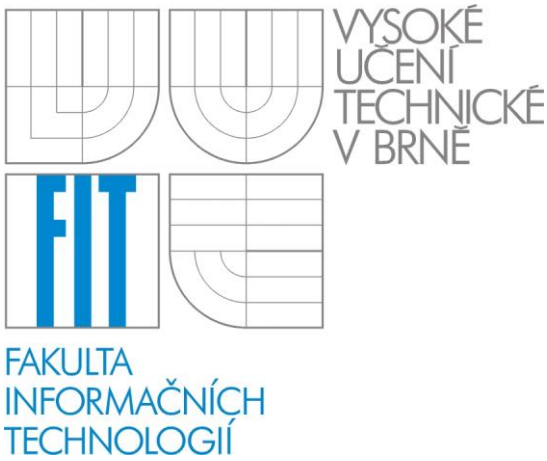

Multimédia

Formát JPEG

Jakub Pelikán, xpelik14
Petr Vizina, xvizin00

7. května 2016

1 Úvod

Cílem projektu je prostudování obrazového formátu JPEG (ITU-T T.81) a následná implementace dekodéru obrazového souboru dostupného v tomto formátu. Vstupní soubor JPEG v režimu baseline (8 bitů/vzorek, sekvenční přenos, Huffmanovo kódování). Další částí je poté uložení dekódovaného obrázku v jiném vhodném formátu (bmp nebo png).

2 Implementace

Postup dekomprese:

- Dekódování bloků obrazu
- Zig-Zag (linearizace obrazu)
- Inverzní diskrétní kosinova transformace
- Subsampling
- Převod barevného modelu YCbCr do RGB
- Uložení do BMP

2.1 Dekódování složky JPEG

Formát JPEG je identifikován pomocí prvních 16 bitů, které odpovídají hodnotě 0xFF, 0xD8. Poté následují jednotlivé obsahující informace o obrázku, data potřebná k jeho dekódování a samotná zakódovaná data.

Bloky:

Začátek bloku je identifikován 0xFF, následováno jednoznačným identifikátorem bloku ve tvaru 0xXX a poté 16-bitovou hodnotou udávající délku daného bloku.

SOF0 - Start Of Frame (0xFF,0xC0)

Indikuje, že dekódovaný obraz je v režimu baseline. Nese informace o přesnosti, rozměru obrázku zadaném jako počet řádků a počet vzorků na řádek. Dále obsahuje počet komponent (YCbCr) a pro každou komponentu její identifikátor, horizontální a vertikální pod vzorkování a identifikátor kvantizační tabulky pro danou složku.

DHT - Define Huffman Table (0xFF,0xC4)

Blok obsahuje huffmanovu tabulku pro dekódování jedné ze složek obrazu. Na začátku bloku identifikace třídy, zda se jedná o huff. tabulku pro AC nebo DC složku, a

poté identifikátor komponenty, která se pomocí tabulky bude dekódovat. Tabulka obsahuje 16 řádku, kde n-tý řádek obsahuje symboly, které jsou zakódovány právě na n bitů.

První položka v tabulce, je umístěna na řádku n a je reprezentována kódem n*0, kód reprezentující následující položku získáme jako kód předcházející položky inkrementovaný o jedna a doplnění doplněný zprava tolika nulami, o kolik řádků níž v tabulce je následující symbol umístěn.

Po získání kódů reprezentující jednotlivé symboly je možno vytvořit huffmanův strom, kde jeden syn reprezentuje 0-bit a druhý syn 1-bit, posloupnost uzlů ve stromu reprezentuje kód z tabulky a listový uzel hodnotu kódu. Pomocí tohoto stromu se čtou hodnoty ze vstupu, a ve chvíli kdy dorazíme ve stromu k listovému uzlu, získáváme dekódovanou hodnotu jako hodnotu listu, a další znak je dekódován opět od kořene.

DOT - Define Quantization Table (0xFF, 0xDB)

Blok obsahuje id kvantizační tabulky a poté matici 8*8 hodnot.

SOS - Start Of Scan (0xFF, 0xDA)

Blok obsahuje hlavičku s informací o počtu komponent obrazu a informace o tom, která komponenta bude dekódována kterou AC a DC tabulkou.

Po hlavičce následují data reprezentující matice 8*8 s daty jednotlivých komponent. První hodnota matice reprezentuje DC složku a získá se jako hodnota součet předchozí DC složky a nové DC složky. Kdy pomocí huffmanova stromu pro DC složku získáme hodnotu, která udává, kolik následujících bitů reprezentuje novou hodnotu DC složky. Poté načítáme 63 AC složek. Opět získáme pomocí huffmanova stromu hodnotu o všem. Počet bitů, které máme, načíst reprezentuje pouze spodní 4-bity a horní 4-bity udávají počet nul, které je potřeba do matice doplnit před získanou AC hodnotu. Pokud je hodnota získaná z huffmanova stromu 0x00, poté je zbytek matice nulový.

Na získanou matici 8*8 poté aplikuje kvantizační matici, a to tak že i-tý prvek získané matice vynásobíme z i-tým prvkem kvantizační matice.

Popsaným způsobem postupně získáme vzorky všech komponent v poměru udaném pod vzorkováním. Například pro 4:1:1 načteme nejdříve čtyři matice 8*8 pro první komponentu poté jednu matici 8*8 pro druhou a jednu matici pro třetí komponentu. Načítáme vzorky jednotlivých komponent až do konce bloku, kde blok je doplněn jedničkovými bity tak aby byl zarovnán na násobek bajtu. Poté je následován blokem End Of Image.

EOF - End Of Image (0xFF,0xF9)

Blog neobsahuje žádná data, pouze signalizuje konec obrázku.

2.2 Zig-Zag

Získané bloky jpeg obrazu jsou linearizovány pomocí metody Zig-Zag, tak aby byla ve 2D matici získána sousednost. Je proto potřeba bloky převést zpět do původního pořadí. K tomu je zapotřebí provést reverzní Zig-Zag. Z důvodu rychlosti převodu bylo vytvořeno konstantní pole, které obsahuje hodnoty nových indexů jednotlivých prvků. Původním indexem se provede indexace do pole a zjistí se tak nový index tohoto prvku.

2.3 Inverzní diskrétní kosinova transformace

Za účelem snížení nenulových prvků jednotlivých 8x8 bloků obrazu, je proveden převod jednotlivých složek do frekvenčního spektra pomocí diskrétní kosinovy transformace. Jedná se o operaci, kde dochází ke ztrátě informací o obrazu. Pro jeho dekódování existuje její inverzní verze. V projektu byla za účelem dekódování implementována následující inverzní diskrétní kosinova transformace:

$$s_{yx} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u \, C_v \, S_{vu} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

Rovnice 1 Inverzní diskrétní kosinova transformace.

$$C_u, C_v = 1/\sqrt{2} \text{ for } u, v = 0$$

$$C_u, C_v = 1 \text{ otherwise}$$

Rovnice 2 Výpočet koeficientů pro IDCT.

2.4 Subsampling

Za účelem redukce výsledné velikosti JPEG souboru je provedena redukce chromatických komponent obrazu. V praxi se nejčastěji využívá:

- **4:4:4**, zde jsou zachovány všechny chromatické a jasové bloky kódovaného obrazu. Matice 8x8.
- **4:2:2**, pro jednu chromatickou odpovídají dvě horizontálně po sobě jdoucí jasové složky. Uspořádání v matici 16x8.
- **4:2:0**, pro jednu chromatickou složku odpovídají čtyři jasové složky, uspořádané do matice 16x16.

V projektu je implementována podpora pro dekódování uvedených typů subsamplingu. Ta je provedena tak, že podle zjištěného typu použitého samplingu se blok přepočítá z rozměru 8x8, na požadovanou velikost.

Přepočítání na 16x8 probíhá tak, že se v horizontálním směru provede suma 4 okolních bodů a vydělí se jejich počtem, tedy zprůměrování. Pro zpracování se tedy použijí 2 body ležící vlevo od aktuálně zpracovávaného bodu a 2 další ležící napravo.

Pro verzi, kdy je potřeba přepočítání z 8x8 do 16x16, se prování prvně horizontální zprůměrování a poté podobným principem ve vertikální rovině.

Získané matice se poté umístí do cílového bufferu. Pro sampling 4:4:4, se neprování žádné přepočítání.

2.5 Převod YCbCr do RGB

Převod z RGB do YCbCr se prování k získání jasové a dvou chromatických složek obrazu, na které lidské oko není tak citlivé a je možné na nich provést větší kompresi. Pro převod mezi těmito modely existuje převodní vzorec. V projektu bylo potřeba provést převod z modelu YCbCr do RGB, k tomu byl využit následující vztah:

$$\begin{aligned} R &= Y + 1.402 \cdot(C_R - 128) \\ G &= Y - 0.34414 \cdot(C_B - 128) - 0.71414 \cdot(C_R - 128) \\ B &= Y + 1.772 \cdot(C_B - 128) \end{aligned}$$

Rovnice 3 Převod YCbCr do RGB

2.6 Uložení do BMP

Převodem do RGB jsme již získali dekódované obrazové data původního JPEGu. Pro uložení dat jsme se rozhodli použít formát BMP. Pro jeho jednoduchost, zde není potřeba provádět kompresi, postačuje vyplnění hlavičky formátu a poté uložení dat. BMP obsahuje hlavičky dvě:

- BITMAPFILEHEADER: hlavička BMP souboru, délka 14bytů
- BITMAPINFOHEADER: informační hlavička o obrázku, délka 40bytů

V programu dochází k vyplnění pouze nezbytných částí obou hlaviček. Vyplněny jsou identifikátor formátu BMP, rozměr obrázku (výška obrázku je uložena jako negativní, kvůli ukládání shora dolů), počet bitových rovin je 1 a počet bitů na pixel 24, komprimace poté žádná. Ostatní hodnoty jsou nastaveny na 0. Do souboru se provede jako první zápis těchto dvou hlaviček a poté již následují data, která se ukládají pixel po pixelu ve formátu (BGR), každá složka je pixelu je uložena na osmi bitech.

3 Obsluha programu

3.1 Překlad a spuštění

Pro implementaci byl zvolen jazyk C++. Program obsahuje řadu zdrojových a hlavičkových souborů. Pro jejich překlad je tedy přibalen soubor Makefile. Na linuxu je poté možný překlad pomocí příkazu „`make`“, výsledkem je binární soubor „mul“. Spuštění programu je poté možné provést následně,

```
./mul input.jpg output.bmp
```

kde „input.jpg“ a „output.bmp“ jsou dva povinné parametry programu. Jako volitelné jsou poté parametry:

- `-bw` : provede převod vstupního obrázku do stupně šedi
- `-qt=8` : zadaná hodnota int, určuje rekonstrukci DCT se zadaným počtem bázových funkcí

Spuštění programu se všemi parametry by vypadalo následovně:

```
./mul input.jpg output.bmp -bw -qt=8
```

Pořadí dvou prvních parametrů je potřeba zachovat, volitelné parametry poté mohou být v různém pořadí.

V archivu ve složde „input“ jsou zahrnuty také testovací JPEG obrázky,

- lena_420.jpg
- lena_422.jpg
- lena_444.jpg

kde číselné označení značí režim subsamplingu souboru.

3.2 Automatické testy

Přibalený Makefile kromě překladu nabízí i sadu základních testů, jež se provádí nad dodanými obrázky. Spuštění testů je možné provést pomocí „`make tests`“. Dojde k provedení 7 předdefinovaných tesů, jejichž výstup bude umístěn do složky „outputs“. Každý test je provázen také výpisem jeho funkce.

4 Závěr a zhodnocení

V projektu se povedlo implementovat program, jenž je schopen dekódovat obrázek JPEG. Omezením je dekódování obrázků o rozměrech, jejichž výška v pixelech a šířka jsou dělitelné osmičkou, z důvodu fixního 8x8 okna. Korektně dekódovány jsou soubory, jejichž subsampling je 4:4:4, 4:2:2 a 4:2:0. Jako rozšíření jsou implementovány volitelné přepínače, jeden pro převod obrázku do režimu grayscale a druhý pro nastavení počtu bázových funkcí pro rekonstrukci obrazu pomocí IDCT. Cílový soubor je poté uložen jako formát BMP.

5 Literatura

[1] CCITT: T.81, *„Information technology – digital compression and coding of continuous - tone still images – requirements and guidelines“* [online]. 1993 [cit. 2016-05-07]. Dostupné na URL: <https://www.w3.org/Graphics/JPEG/itu-t81.pdf>

[2] Wikipedie: Otevřená encyklopedie: *„YCbCr“* [online]. 2016, aktualizováno 2016-04-30 [cit. 2016-05-07]. Dostupné na URL: <https://en.wikipedia.org/wiki/YCbCr>

[3] Root.cz: Informace nejen ze světa linuxu: *„Grafický formát BMP – používaný a přitom neoblíbený“* [online]. 2006, aktualizováno 2006-10-19 [cit. 2016-05-07]. Dostupné na URL: <http://www.root.cz/clanky/graficky-format-bmp-pouzivany-a-pritom-neoblibeny/#k05>

[4] ImpulseAdventure: *„JPEG Huffman Coding“* [online]. 2007, aktualizováno 2009-09-22 [cit. 2015-05-12]. Dostupné na URL: <http://www.impulseadventure.com/photo/jpeg-huffman-coding.html>

[5] ImpulseAdventure: *„Designing a JPEG Decoder“* [online]. 2008 [cit. 2015-05-12]. Dostupné na URL: <http://www.impulseadventure.com/photo/jpeg-decoder.html>