KKO dokumentace

Lukáš Plevač - xpleva07

May 9, 2024

1 Rozbor problému

Cílem projektu do KKO je navrhnout bezeztrátový komprimační algoritmus, který bude sloužit pro komprimaci surových šedotónových obrázků. Pro kódování dat má být použit huffmanův kód. Pro dosažení vhodného komprimačního poměru je samotné kódování předcházeno vhodným skenováním obrazu následovaného vhodným modelem a RLE. Tyto části mají za úkol zmenšit abecedu, která bude kódována, ale také především zajistit častou existenci nějakého symbolu a pokusit se existenci ostatních symbolů naopak zmenšit. Díky tomu bude moci být tento znak kódován malým počet bitů. RLE sloučí poté na zmenšení počtu celkového počtu symbolů, tak že repetice symbolů vhodně zakóduje. Protože na dekódování obrazu budeme potřebovat informace, které na straně dekodéru nejsou dostupné bude nutné do souboru navíc zakomponovat hlavičky, které tyto informace přenesou na stranu dekodéru.

2 Implementace

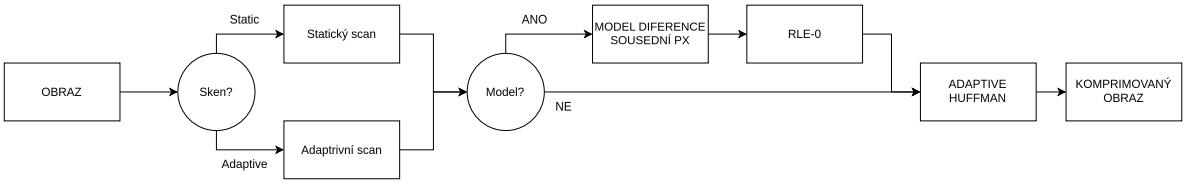

Figure 1: Schéma komprimace

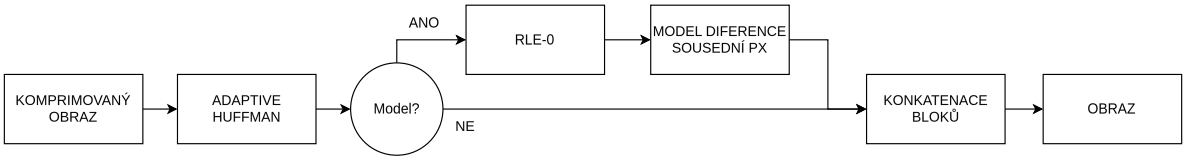

Figure 2: Schéma dekomprimace

2.1 Skenování obrazu

VSTUP: načtený obrázek ze vstupu
VÝSTUP: vektor serializovaných bloků

Skenování slouží k vhodné serializaci 2D vstupních dat.

2.1.1 Statické

Statické skenování serializuje 2D obraz, tak že jednotlivé řádky obrazu naskládá za sebe. Protože je obraz již tímto způsobem uložen v vstupním souboru může toto skenování jen předat vtupní data dále.

2.1.2 Adaptivní

Adaptivní skenování rozloží vstupní obraz do menších 16x16 bloků, tyto blocky mohlou být serializovány po řádcích nebo po sloupcích. Směr serializace se zvolí tak, aby se maximalizoval počet schodných číslic za sebou. Informace o zvoleném směru skenu je uložena do struktury k samotnému skenu.

2.2 Model a RLE-0

VSTUP: vektor serializovaných bloků
VÝSTUP: vektor serializovaných bloků

Cílem následující části je repetice stejných hodnot změnit na repetice nul a ty následně pomocí RLE-0 zkomprimovat na menší velikost.

2.2.1 Model

Jako model byla použita diference sousedních pixelů. Tato metoda zajisté že řepeteice stejných symbolů budou přepsány na repetice nul. Model funguje tak že od současného pixelu je odečtena hodnota pixelu předcházející. Pokud serializace v předchozím kroku proběhla správně mel by tento krok vygenerovat řetězce nul, které budou místy protkány jinými čísly. Svůj výstup uloží místo starých dat skenu.

2.2.2 RLE-0

Tato metoda slouží k zredukování počtu nul v blocích. Tato metoda má oproti standartnímu RLE výhu v tom že nikdy neprovede expazi vtupu, proto není v této části kontrolovat zda opravdu došlo k zmešení bloku. Nevýhodou této metody, je to že vyžaduje přidání jednoho nového symbolu do abecedy, a kvůli tomu se nám již jeden pixel nevejde do 8b. Nicméně protože pro kódování používáme huffmanův adaptivní kód není tento problém problémem velkým. Svůj výstup uloží do poitru na RLE data.

2.3 Kodér

VSTUP: vektor serializovaných bloků
VÝSTUP: string zakódovaných bitů

kodér slouží pro transformaci dat na binární reprezentaci, která bude moci být uložena do souboru.

2.3.1 Formát souboru

Protože dekodér pro svoji funkčnost potřebuje informace, které zdá je kodér je nutné tyto data předat pomocí zakódovaných data. Těmito daty jsou velikosti obrázku, počty bloků na řádek a sloupec, které jdou použít k zjistění kolik dat má být přijato. Dále je v zakódovaných datech uložena informace u každého bloku jakým způsobem byl skenován a zda byl zkomprimován nebo zda následující hodnoty nejosu komprimovány. Data také nesou informaci jakým způsobem byl skenován obrázek.

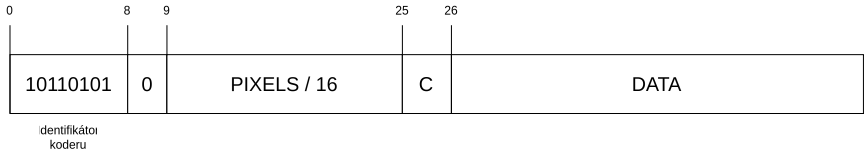

Figure 3: Schéma souboru statického skenování

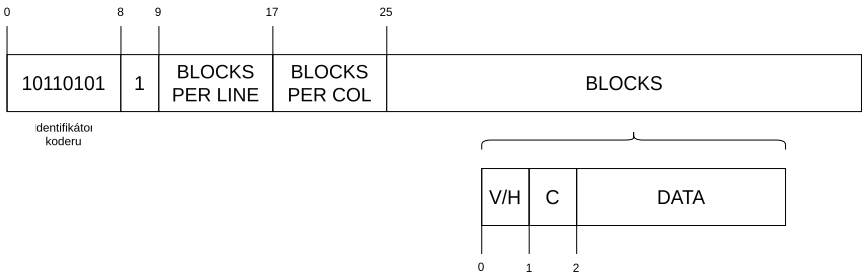

Figure 4: Schéma souboru adaptivního skenování

2.3.2 Adaptivní huffmanův kód

Samtné kódování dat je zajištěno pomocí adaptivního hummanova stromu. Pro každý block je vytvářen strom nový a pomocí toho stromu je zakódován celý blok, pokud by zakódovaný výstup byl delší než původní data bez RLE bude nastaven flag C u bloku na 0 a data budou zkopírována jako osmibitové surové hodnoty bez jakékoliv komprimace.

Nutnou poznámkou je, že u dekódování je při zapnutém RLE nutné aktivně dekódovat RLE data přímo z dekódovaného streamu, aby bylo možné zjistit zda již byl přečten celý blok, protože blok nemá žádnou ukončovací značku, ale pouze danou velikost.

3 Výsledky

Následující výsledky byly změřeny na testovací sadě, na procesoru Intel(R) Celeron(R) CPU N2940 @ 1.83GHz.

parametry	průměrná doba komprimace	průměrná doba dekomprimace	průměrně bitů na pixel
	19,36s	4,79s	5,74
-m	9,82s	8,45s	2,82
-a	2,04s	4,02s	4,85
-a -m	1,34s	3,13s	3,23

Table 1: Průměrné výsledky

obraz		-m	-a	-a -m	entropie zdroje
df1h.raw	8 bpp	1 bpp	4,61 bpp	0,48 bpp	8 bpp
df1hvx.raw	4,58 bpp	1,23 bpp	3,72 bpp	1,16 bpp	4,51 bpp
df1v.raw	8 bpp	0,02 bpp	4,61 bpp	0,48 bpp	8 bpp
hd01.raw	3,88 bpp	2,67 bpp	4,04 bpp	3,43 bpp	3,83 bpp
hd02.raw	3,710 bpp	2,61 bpp	3,88 bpp	3,33 bpp	3,64 bpp
hd07.raw	5,61 bpp	3,31 bpp	4,811 bpp	3,66 bpp	5,58 bpp
hd08.raw	4,24 bpp	2,92 bpp	4,29 bpp	3,37 bpp	4,21 bpp
hd09.raw	6,66 bpp	4,55 bpp	6,04 bpp	4,97 bpp	6,62 bpp
hd12.raw	6,20 bpp	3,82 bpp	5,31 bpp	4,24 bpp	6,17 bpp
nk01.raw	6,51 bpp	6,05 bpp	7,22 bpp	7,12 bpp	6,47 bpp

Table 2: Efektivnost komprimace

obraz		-m	-a	-a -m
df1h.raw	24854	159	703	136
df1hvx.raw	3814	7309	683	423
df1v.raw	17049	22	702	136
hd01.raw	25954	9997	2369	1716
hd02.raw	26001	9268	2323	1747
hd07.raw	22568	10336	2165	1246
hd08.raw	6729	6558	1813	1342
hd09.raw	21470	20663	2969	1816
hd12.raw	24070	12956	2855	1636
nk01.raw	21095	20953	3770	3298

Table 3: Doba komprimace v ms