VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

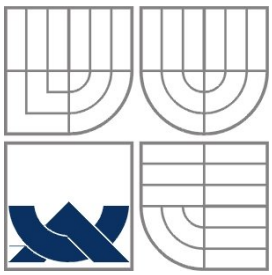
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PREHLIADAČ WSQ OBRÁZKOV POD WINDOWS

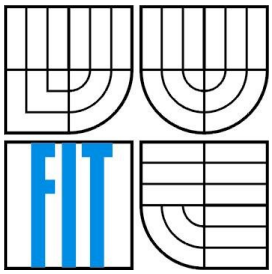
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE Zoltán Kovács
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PROHLÍŽEČ WSQ OBRÁZKŮ POD WINDOWS

WSQ VIEWER UNDER WINDOWS PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Zoltán Kovács
AUTHOR

VEDOUCÍ PRÁCE — Drahanský Martin, Ing., Ph.D.
SUPERVISOR

BRNO 2007

Abstrakt

Cílem bakalářské práce je návrh a realizace prohlížeče obrázků otisků prstů, které jsou uloženy ve formátu WSQ (Wavelet Scalar Quantization). Čtenář se obeznámí se základním rozdělením typů obrázků a také se způsoby komprese. Dozví se základy o vlnkové transformaci a o způsobu její aplikace v kompresi obrazových dat. Bude mít přehled o formátu JPEG2000 a formátu WSQ. Obeznámí se taky s principem implementace prohlížeče a možnostmi které nabízí. Celková funkce prohlížeče musí umožnit uživateli vybrat si obrázek, na kterém se provede komprese, zobrazí se otisk prstu a následně se dá uložit tento obrázek do některého známého obrázkového formátu.

Klíčová slova

WSQ, prohlížeč obrázků, obrázkový konvertor

Abstract

The object of this bachelor project is proposal and realization of a picture viewer for viewing fingerprint images compressed with the WSQ (Wavelet Scalar Quantization) algorithm.
The reader will get to know the basic division of picture types and basic methods of compression. Also, the reader will gain an overview of the JPEG2000 and WSQ formats, familiarize with the implementation of the viewer and with the functions the viewer offers.
The viewer enables to select a compressed image which is than processed and the fingerprint image is displayed. This image can be saved to a supported image format.

Keywords

WSQ, image viewer, image converter

Citace

Zoltán Kovács: Prehliadač WSQ obrázkov pod Windows, bakalárska práca, Brno, FIT VUT v Brne, 2007

Prehliadač WSQ obrázkov pod Windows

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Martina Drahanského, Ph.D.
Další informace ohledne MFC mi poskytl Marián Krivda.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Zoltán Kovács
14. 5. 2007

Poděkování

Děkuji školiteli Ing. Martinovi Drahanskému, Ph.D, za poskytnuté materiály ze kterých jsem mohl čerpat námět i inspiraci při psaní bakalářské práce, a také za pomoc kterou mi poskytl při řešení práce

Chci taky poděkovat Mariánovi Krivdovi, který mi poskytl odbornou pomoc ve vytvoření grafické nadstavby bakalářské práce.

© Zoltán Kovács, 2007.

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod

Digitálne spracovanie obrazu. Stretávame sa s týmto pojmom deň čo deň. Je prítomná ak surfujeme na webe, ak si zapneme mobilný telefón ak si vytlačíme na tlačiarni fotky, ktoré sme si urobili na výlete. Tieto informácie sú niekedy cenné, inokedy zanedbateľné, ale predsa sa stali potrebnými pre náš život. Potrebujeme tieto informácie spracovať, musíme mať možnosť ich ukladať, prenášať a ak je potrebné, tak ich aj zobraziť. Musíme na to všetko mať prostriedky. Potrebujeme prostriedok ktorí zabezpečí, že tieto informácie budú zachytené korektne, potrebujeme prostriedok, ktorý je nosičom týchto informácií a taktiež potrebujeme prostriedok po ktorý siahneme ak chceme vidieť tieto informácie.

V súčasnosti existuje mnoho spôsobov digitalizácie obrázkov. Od digitálnych fotoaparátov cez čítačky dokumentov po rôzne senzory. Tieto informácie sa väčšinou zhŕňajú do rady pixlov a sú zobrazené buď na monitore alebo na papieri. Táto hromada pixlov je však veľmi nemotorná a veľká na to, aby sa ukladala pre neskoršie použitie. A práve preto existujú kompresné formáty, ktoré tieto hromadu pixlov zahustí do menšej hromádky, ktorá sa uloží na príslušné médium. Potrebujeme zabezpečiť, aby táto hromádka bola spätne znovu poskladateľná do tvaru zrozumiteľného užívateľovi.

Náplňou mojej práce je však o trochu špecifickejší projekt, a to prehliadač obrázkov WSQ. Potrebujeme si najprv vyjasniť niekoľko základných informácií čo je to vlastne WSQ. WSQ znamená wavelet scalar quantization, tento pojem bude vyjasnený v neskoršej časti dokumentácie, zatiaľ stačí ak si vyjasníme, že sa jedná o formát uloženia obrázkov, ale nie hocijakých obrázkov. Jedná sa totiž o špecializovaný formát vyvinutý pre otlačky prstov. Tento formát je už pomerne starý, nakoľko sa jeho vývoj spojuje s jednou najväčšou organizáciou, a to Federálny Úrad pre Vyšetrovanie známy pod skratkou FBI. Tento Úrad mal k dispozícii dostatočné finančné prostriedky, aby si mohol dovoliť vyvinúť formát, ktorý by bol schopný ukladania odtlačkov prstov kriminálnikov na území USA. Vývoj tohto formátu stáli nemalé peniaze, ale stálo to za to. Tento formát pomohol FBI v zjednodušení dokazovania viny, hľadaní kriminálnikov a celý proces sa tak výrazne skrátil.

Rapídnym rozšírením počítačov sa stali obrázky ľudom prístupné a tak sa niektorí začali zaujímať aj o špeciálne formáty, ako je aj WSQ. Cez všetky snahy sa tento formát nedostáva do komerčného behu FBI si ho dlho privlastňovala a informácie pre verejnosť vôbec neboli dostupné. Až v posledných rokoch začali prejavovať záujem o využitie odtlačkov prstov pre použitie zabezpečovacích systémov aj pre veľkú verejnosť, a nie len pre mamutie firmy, ktoré potrebovali vysokú mieru bezpečnosti.

Bol teda potrebný vývoj softwaru, ktorý dokáže pracovať s týmto formátom, ponúkať komfortné prehliadanie obrázkov, umožňovať zobrazované dáta ako celok. Užívateľ sa teda oboznámi v nasledujúcich častiach ako prebieha kompresia tohto obrázku a oboznámi sa taktiež s návrhom prehliadača.

2 Obrázky

V súčasnosti sa s týmto pojmom stretávame deň čo deň. Sú prítomné všade a v rôznej forme. Stretávame sa s nimi na papieri vo vytlačenej forme, na obrazovkách TV, na monitoroch počítačov či už v analógovej alebo digitálnej forme. Sú nosičom informácií. Zachovávajú spomienky, niekedy dôležité dáta, inokedy sú len výtvorom výmyslov. Existujú dva typy obrázkov. Vektorové, ktoré sa skladajú z jednoduchých geometrických objektov a bitmapových, ktoré tvoria pixli ukladané vedľa seba. V tejto dokumentácii sa budem zaoberať výhradne obrázkami bitmapovými, takže pojem obrázok je myslený ako bitmapový obrázok.

2.1 História

Počítače v 60.-70. rokoch minulého storočia používali zobrazovaciu jednotku, ktorá zabezpečovala prezentáciu dát v znakovej forme, teda zobrazovali sa len reťazce. Existovala síce možnosť zobrazovania obrazcov, napríklad vykresľovanie čiar a iných objektov, ale tento spôsob nebol možný uchovávať realistické obrázky. V 80. rokoch sa začali vyvíjať grafické nadstavby, hardware začal dosahovať takú úroveň, že sa mohla odštartovať éra počítačovej grafiky. Postupne sa zvyšovali aj kapacity diskov, potrebné na ukladanie dát. Taktiež sa zvyšovali rozlíšenia monitorov, objavovali sa farebné tlačiarne a scannery. Vyvíjalo sa realistickejšie zobrazovanie, ktoré umožňovalo uchovávať a zobrazovať obrázky v digitálnej forme.

2.2 Zobrazovanie

Zobrazovanie obrázkov však prináša niekoľko problémov. Tým najhlavnejším problémom je, že ako uložiť obrázok na disk alebo na iný nosič dát tak, aby sme dokázali znovu načítať jeho obsah a korektne zobraziť. Môžeme uvažovať nad tým, či potrebujeme k obrázku rýchly prístup, teda nenáročný, alebo žiadny algoritmus ukladania a načítania za cenu, že obrázok bude mať väčšiu veľkosť alebo naopak, teda potrebujeme šetriť miesto i za cenu náročnejšieho výpočtu. Zakódovanie obrázku sa nazýva kompresia. Musíme pravdaže brať aj do úvahy, či potrebujeme obrázok znázorňovať ako bezstratový, teda tak, že po zakódovaním a následným dekódovaním daným algoritmom, dostávame pôvodný obrázok alebo stratový, čo znamená, že pôvodný obrázok sa môže zmeniť. Existuje mnoho formátov uloženia obrázkov.

2.2.1 Bez kompresie

Formáty bez kompresie sú formáty, ktoré ukladajú postupnosť pixlov na obrazovke do postupnosti bajtov. Nie sú použité žiadne algoritmy, jediné čo sa vyberie je reprezentácia dát v postupnosti bajtov. V dnešnej dobe zobrazovacie jednotky pracujú na základe RGB zložiek. Tieto zložky udávajú pomer farieb červenej, zelenej a modrej na použitom pixli. Pomer týchto troch farieb udáva výslednú farbu na danom pixli. Takto sa tvorí farebný obrázok. Môžeme vytvoriť taktiež čiernobiely obrázok, pri ktorom sa udáva len jeden bit na jeden pixel. Tento bit je nastavený podľa toho, či je daný pixel čierny alebo biely. Existuje aj obrázok, odtiene šedej. Takýto obrázok sa tvorí tak, že sa ukladá do bajtu číslo, nakoľko má byť pixel sivý. Toto číslo sa rozloží do RGB zložky rovnomerne, čo zabezpečuje, že výsledok je odtiene šedej.

Pri používaní formátu bez kompresie je použitý BMP. Tento formát je známy a rozšírený, ľahko sa s ním pracuje. Používa ho najmä operačný systém Microsoft Windows. Existuje niekoľko verzií, ale základný princíp je nezmenený. Väčšina verzií sa rozdeľuje len v ukladaní metadát. Jednoducho sa môžeme dostať k obsahu takého súboru, ale na druhej strane platíme za to miestom na disku. Veľkosť BMP súboru je závislá na použitej bitovej hĺbke, ktorá udáva presne koľko bitov bude udávať farbu. BMP obrázky majú neraz rozmer niekoľkých megabajtov. Najčastejšie sa používajú 1 bitové, 8 bitové, 24 bitové a 32 bitové formáty. 1 bitová farebná hĺbka sa používa pre čiernobiele obrázky, 8 bitová farebná hĺbka sa používa pre šedé, 24 a 32 bitové pre farebné obrázky, kde každá RGB zložka je zobrazená na 8 bitoch.

Formát BMP podporuje aj jednoduchú kompresiu.

2.2.2 Kompresie bezstratové

Hlavnou myšlienkou bezstratovej kompresie je uložiť obrázok na čo najmenšie miesto na médium s podmienkou, že sa dá zrekonštruovať spätne pôvodný obrázok. Niektoré využívajú kompresné formáty, ktoré sú používané pri kompresii iných dát, iné majú špecializované algoritmy. Pri obrázkoch sa niekedy používa aj kódovanie po sebe nasledujúcich identických pixlov do niekoľkých znakov. Výhodou takýchto algoritmov je stopercentné zachovanie informácií. Pri týchto formátoch miera veľkosti obrázku záleží na charakteru obrázku. Ak sa v obrázku vyskytujú súvislé plochy alebo identické farby, miera kompresie je oveľa väčšia ako pri obrázku s rôznorodým rozložením farieb.

Formáty bezstratové sú napríklad PNG (Portable Network Graphics), GIF (Graphics Interchange Format) či TIFF (Tag Image File Format).

2.2.2.1 GIF

GIF (Graphics Interchange Format) je grafický formát s využitím kompresie Lempel-Ziv-Welch. Podporuje maximálne 8-bitové obrázky teda 256 farieb alebo 256 odtieňov šedej. Má dve verzie 0.87 a 0.89a. GIF umožňuje ukladať animované gify, čo sú jednotlivé obrázky zobrazované v určitých intervaloch. Tento formát sa veľmi rozšíril aj vďaka internetu, pretože umožňoval postupné zobrazovanie obrázku už po načítaní 1/8 dát, zatiaľ čo iné formáty potrebovali na zobrazenie načítať celý súbor.[1]

2.2.2.2 TIFF

Tento formát je príkladom veľmi univerzálneho grafického formátu, čo však prináša veľkú zložitosť a odlišnosť pri načítavaní tohto formátu. Spolu s formátom JPEG (Joint Photographic Experts Group) a PNG je medzinárodným štandardom pre kódovanie statických obrazov. Formát prešiel pomerne veľkým vývojom. Veľká škálovateľnosť spôsobuje, že len málo programov dokáže zobrazovať všetky povolené varianty tohto grafického formátu.[1]

2.2.2.3 PNG

Formát vyvinutý konzorciom w3c a doporučený ako náhrada GIF-u. Má všetky vlastnosti ako GIF, okrem animácie, ale je novší, modernejší. Ten istý obraz uložený ako PNG je menší ako GIF, pretože PNG používa kvalitnejší algoritmus nestratovej kompresie LZ77 a Huffmanovú kompresiu ako je LZW pri GIF–e. [2]

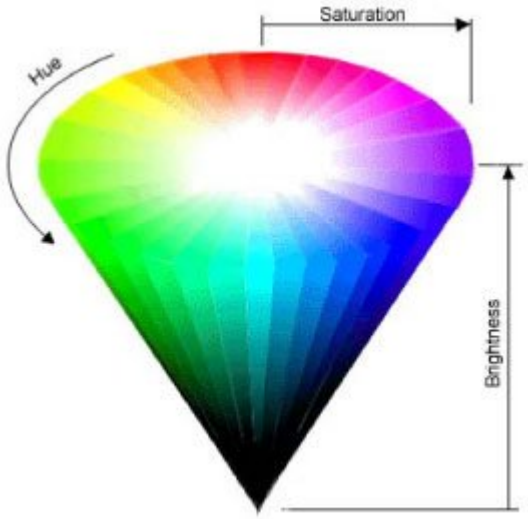
2.2.3 Kompresie stratové

Princíp stratovej kompresie spočíva v tom, že už po uložení obrázku a následným zobrazením, nedostávame ten istý obrázok ale len obrázok ktorý sa podobá na pôvodný. Pritom sa berie do úvahy nedokonalosť ľudského zraku, a tak sa nám zdá, že obrázok je takmer identický s pôvodným obrázkom. U stratových formátoch treba brať do úvahy mieru kompresie. Musíme tu spraviť kompromis, medzi veľkosťou obrázku a mierou straty informácií. Je potrebné nejako vymedziť z obrázku informácie dôležité a menej dôležité, teda také, ktoré sa môžu stratiť pri kompresii. Existuje niekoľko kompresných metód. Najznámejší a najpoužívanejší je JPEG, ďalej existuje takzvaná Fraktálová kompresia a JPEG2000, s ktorou sa budem zaoberať v ďalšej kapitole.

2.2.3.1 JPEG

Experimenty dokázali, že ľudské oko je výrazne citlivejšie na svetlosť farby, ako na farebný odtieň. Práve toto využíva aj jpeg. Pre pochopenia princípu skúsme nasledujúci postup. Ak pretransformujeme obrázok z farebného systému RGB do farebného systému HSB (Obrázok 1) tak

prieskumy ukazujú, že na osi B (Brightness), teda svetlosť je oko výraznejšie citlivé, teda uchováva viac užitočných dát ako ostatné dve osi. Preto sa osi H a S zjednodušujú, a tu sa stráca zhruba 50% informácií, nakoľko sa dve vedľajšie odtiene zlučujú do jedného odtieňa.



Obrázok 1: Farebný model HSB

(zdroj: http://www.cecs.csulb.edu/~jewett/colors/hsb.jpg)

JPEG nepracuje s celým obrazom, len s časťami 8x8 pixel. Táto vlastnosť má historické dôvody, nakoľko JPEG bol vyvinutý koncom 80. rokoch minulého storočia a vtedy sa bral veľký ohľad na efektivitu výpočtu. Ak vynásobíme 8x8 dostávame 64 čo je taktiež násobkom dvojky a teda výpočet algoritmu mohol nabrať dostatočnú efektivitu. Táto vlastnosť však v súčasnosti spôsobuje často problémy.

Formát JPEG používa diskrétnu cosínusovú transformáciu (DCT). Po prevedení tejto transformácie obdržíme kmitočtovú reprezentáciu obrázku, ktorá je významná tým, že kde je nízky kmitočet, tam sa v obrázku vyskytujú súvislé plochy bez zmeny, a kde je vysoký, tam sú prudké zmeny farby. Tieto informácie nám poslúžia ku kvantovaniu, keď môžeme určité zložky zanedbať a ukladať len podstatne detaily. V praxi sa prevádzajú hodnoty farebných koeficientov (teda hodnoty HSB) na frekvenčné údaje. Na základe nich sa vypočítajú koeficienty a stačí ich už len uchovávať. Zatiaľ máme 64 koeficientov a na súradnici (0,0) bloku máme priemernú farbu celého bloku. Koeficienty majú hodnoty v rozpätí -1024 – 1023 čo je 11 bitov. Sila kompresie sa udáva práve na tom, ako sa upravuje tento údaj na kvantovej matici. Niekedy sa udáva len na 8 bitoch. Tieto transformácie spôsobujú práve to, že ak je obrázok plný plynulých prechodov, je pri vysokej miere kompresie ešte stále dobre rozoznateľný ako pri obrázku s prudkými zmenami v prechodoch.

Formát JPEG patrí v súčasnosti k najrozšírenejším formátom, a to aj vďaka internetu. Je veľmi dobrým formátom, avšak je už trochu zastaralým, a má aj niekoľko limitov, napríklad nepodporuje 48 bitovú farebnú hĺbku, alebo rozlíšenie obrázku nad 64000x64000 pixlov.

2.2.3.2 Fraktálová kompresia

Toto je zaujímavý algoritmus, ktorý vychádza z teoretických poznatkov, ktoré tvrdia, že obrazce reálneho života sa periodicky opakujú v rôznych veľkostiach. Táto kompresia je založená na tom, že v obraze hľadajú kúsky obrazu samotného. Neskôr bol algoritmus zefektívnený tak, že sa nehľadá priamo obraz celého obrazu, ale i jeho častí. Z tohoto stručného popisu už vyplývajú niektoré zaujímavé skutočnosti, ako napríklad to, že obraz je možné nekonečne približovať a tým sa v obraze logicky od určitej úrovne objavujú detaily, ktoré v originály neboli. Príjemnou vlastnosťou tejto kompresie je fakt, že sa nikdy nestretávame s klasickými štvorcovými artefaktmi z JPEG-u, pretože fraktály majú ďaleko prirodzenejšie tvary. Budúcnosť tejto zaujímavej kompresie je trochu sporná, pretože sa na kompresiu napríklad fotiek veľmi hodí, ale pre tradičnú geometriu je len ťažko použiteľná. Naviac aj u fotiek je lepší než JPEG až pri vyšších kompresných pomeroch. Posledným, ale zďaleka nie najmenej významným bremenom je licencovanie tejto technológie

3 JPEG2000

Vývoj formátu JPEG2000 sa začal v roku 1995 ako nástupca JPEG-u. Prvá verzia bola dokončená v roku 2000. Odhady boli také, že bude priamym nástupcom JPEG-u a rýchlo sa rozšíri. Skutočnosť však bola iná. JPEG2000 ešte ani v súčasnosti nedosahuje také pomery aké by si zaslúžil. Dôvod bol jednoduchý. Bol chránený patentmi. Ani jednému formátu to neprispieva na popularite. Aj formát gif mal ten istý problém, avšak GIF už v súčasnosti nepodlieha patentom, takže je volne dostupný. JPEG2000 je síce formát dobrý, ale dosť nízka podpora v internetových a obrázkových prehliadačoch, grafických editoroch mu neprispieva.

3.1 JPEG2000 vs. JPEG

Oproti JPEG-u, stavia na vlnkovej transformácie vlnková transformácia znamená, že vzorkovací signál nie je periodický, ale je to vlnka, ktorá príde, zakmitá a následne odíde. Toto prináša so sebou radu možností, ktoré sa dajú efektívne využívať. Jednou z najväčších kladov je to, že sa už nemusí limitovať veľkosť vzorkovanej plochy ako pri JPEG-u na 8x8, ale sa môže navzorkovať celý obrázok ako celok (Obrázok 2) Avšak JPEG2000 podporuje možnosť rozdelenia obrázku do bloku, pre flexibilnejšie spracovane. Táto možnosť je voliteľná a nemusí byť využitá vôbec.

Obrázok 2: Porovnanie obrázku kompresiou JPEG a JPEG200

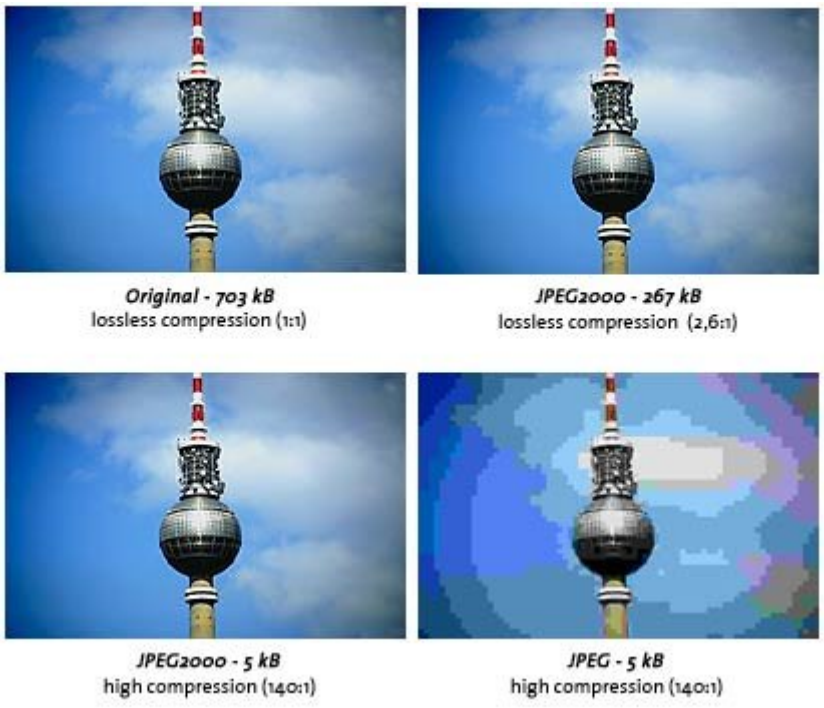
V JPEG2000 je použitý unifikovaný spôsob dekompresie obrázku. Toto je taktiež dôležitá novinka oproti JPEG-u, nakoľko JPEG má 44 dekompresných metód a taký veľký počet je podporovaný len zriedka v prehliadači, či grafickom editori. Väčšinou sa používa len niekoľko dekompozičných metód, ktoré sú vybrané tak, aby bolo možné obrázky zobraziť bez problémov.

Nezanedbateľnou vlastnosťou JPEG2000 je aj to, že kompresia je oveľa rýchlejšia, kvalitnejšia a efektívnejšia. V percentách to znamená zhruba 20-30 percentný nárast stratového kompresného pomeru oproti klasickému JPEG-u, ale v krajných prípadoch môže dosahovať aj 60-70%. Ako JPEG aj JPEG2000 umožňuje kompresiu bezstratovú. Aj tu je výrazný pokrok, teda veľkosť obrázku

bezstratovej kompresie môže byť až o polovicu menší ako pri JPEG. Toto číslo je síce veľmi pekné, ale ešte stále zaostáva oproti bezstratových kompresných formátoch, ako je napríklad PNG alebo TIFF.

JPEG2000 však má v sebe vlastnosť, ktorá mu umožňuje, že obrázok je odolnejší proti chybám v dátach. Keď sa zmenia nejaké bity v obrázku pri manipulácii, algoritmus dekompresie je ešte stále schopný zrekonštruovať obrázok.

Ďalšou dôležitou zmenou oproti JPEG-u je pracovanie s obrázkami pri veľkej miere kompresie. JPEG sa už začína chovať divne pri ukladaní dát nižších ako 0,25 bit na jeden pixel. Neznamená to, že by obrázok nebol viditeľný, ale už prechody farieb sú rozmazané, a začína byť zreteľný blok 8x8 pixlov (Obrázok 3). Pri obrovskej miere kompresie začína byť obrázok nerozoznateľný (Obrázok 4). JPEG2000 sa ešte s takýmito obrázkami vysporiada efektívne.



Obrázok 3: Demonštrácia obrázkov JPEG a JPEG2000 pri vysokej miere kompreise

(zdroj: literatúra 3)

Obrázok 4 : Demonštrácia obrázkov JPEG a JPEG2000 pri vysokej miere kompresie

(zdroj: literatúra 4)

JPEG2000 taktiež má zabudovanú možnosť použitia takzvaných zložených (compound) dokumentov. Jedná sa o také dokumenty, ktoré majú v sebe zabudovanú jednak grafiku a popri nej aj text. JPEG je pre takéto účely nepoužiteľný.

Ako už bolo vyššie spomenuté, JPEG má limitu veľkosti obrázku 64000x64000 pixlov. Táto hranica sa odbúrava v JPEG2000.

JPEG2000 sa taktiež lepšie vysporiada s počítačom tvorenými grafikami, teda s takými obrázkami, ktoré majú v sebe ostré prechody. Pri vývoji JPEG-u bol kladený dôraz na obrázky prirodzeného charakteru.

Ďalším kladom JPEG2000 je, že máme možnosť využiť rôzne farebné módy. JPEG dokázal pracovať len s obrázkom v RGB módu.

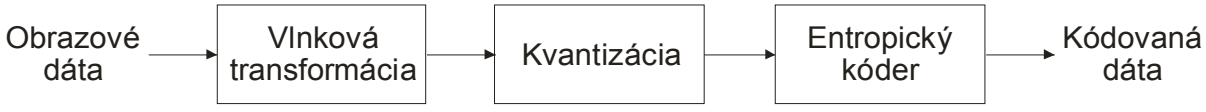
Už pri bezstratovej kompresii sa stretávame s názov progresívna transmisia. Tento pojem znamená to, že obrázok sa zobrazuje už pri časti zapracovaní dát, a postupne sa zaostruje, teda vykresľuje do konečnej podoby. Väčšina JPEG dekodérov ukáže obrázky len pri celom načítaní. Pri JPEG2000 a taktiež aj pri GIF i PNG sme schopní vidieť náčrt obrázku ešte pred načítaním celku. Toto je veľmi užitočné pri internetových aplikáciách, kde pri pomalom pripojení musíme čakať dosť veľa aby sme mali prehľad.

V JPEG2000 dokážeme niektoré oblasti obrázku vyzdvihnúť prioritne. Toto sa realizuje tak, že táto oblasť sa v dátovom toku zasadzuje na prioritné miesto, a tak dostáva vyššiu prioritu. Môže sa tu vyskytnúť aj časť s vyšším rozlíšením.

V JPEG2000 sa stretávame aj s pojmom metadáta. Tento formát dokáže uchovávať v metadátach napríklad meno autora, dátum vytvorenia obrázku, popis obrázku či vlastníka autorských práv.

3.2 Spracovanie JPEG2000 obrázku

Nasledujúci obrázok ukazuje ako sa postupne prevádza kompresia dát.



Obrázok 5: priebeh kompresie obrázku JPEG2000

Zdrojové dáta sa najprv rozložia do komponentov, teda do zložiek farebného modelu, väčšinou RGB. Obraz a jeho komponenty sú rozložené na dlaždice, ktoré sú pravouhlé neprekrývajúce sa časti originálneho obrazu, ktoré sú komprimované nezávisle, ako by to bol celý obraz a je povolená ľubovoľná jeho veľkosť až do veľkosti originálneho obrazu. Dlaždica v komponente je základná jednotka originálneho a rekonštruovaného obrazu.

Vlnková transformácia sa aplikuje na každú dlaždicu a jednotlivé dlaždice sú pomocou vlnkovej transformácie dekomponované do rôznych stupňov rozlíšenia. Dekompozičné stupne vytvárajú subpásma (subband) koeficientov, ktoré popisujú frekvenčné vlastnosti lokálnych oblastí dlaždicových komponentov. Tieto koeficienty sú kvantované a umiestnené do kódového bloku.

Bitový zápis koeficientov kódového bloku je entropicky zakódovaný. Kódovanie môže byť prevedené tiež tak, aby isté výraznejšie časti obrázku boli zakódované vo vyššej kvalite ako okolie.

Značky, ktoré sa pridávajú do bytového záznamu (bit-stream) poskytujú chybovú odolnosť. Konečný kódový záznam má na začiatku hlavičku, ktorá popisuje obraz a rôzne dekompozičné a kódovacie štýly. Tieto informácie sa využívajú pre dekódovanie, rekonštrukciu obrazu s požiadavkou na určité rozlíšenie, a presnosť.

Voliteľný súborový formát popisuje význam obrazu a jeho komponentov v kontextu aplikácie. Kódovanie JPEG 2000 je založené na EBCOT algoritmu (Embedded Block Coding with Optimized Truncation of embedded bit-stream).

4 Vlnková transformácia

Formát JPEG2000 pracuje s obrázkom ako s celkom a prevádza ho na popisy pomocou vlnkových funkcií (wavelet transform). Prevod je niekoľkoprechodový. Počet prechodov určuje kompresný pomer a kvalitu dekomprimovaného obrázku. Čím menej prechodov, tým je vyšší kompresný pomer a tým nižšia kvalita obrázku. Každému prechodu zodpovedá zvláštny dátový blok komprimovaného súboru. Vlnková kompresia sa objavila v polovici 80. rokov. Pôvodne ako matematická metóda. Pomenovanie má po matematičke Ingrid Daubechies (Princeton University), ktorá mala najväčší podiel na vzniku tohoto formátu.

4.1.1 Základný princíp

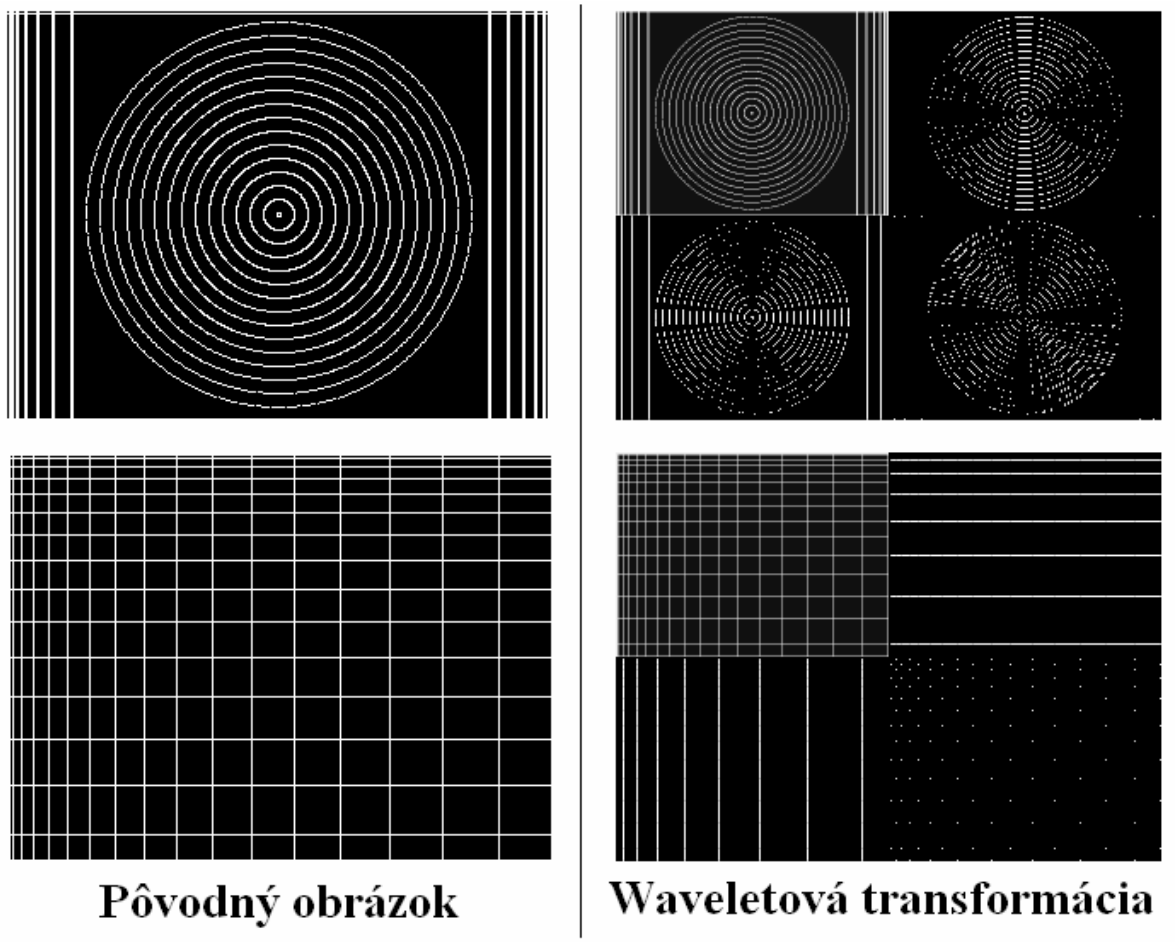
Každá transformačná technika má svoju oblasť pôsobnosti, v ktorej má určité výhody a nevýhody, teda i vlnková transformácia. Vlnková transformácia (VT) je tak isto ako Fourierova transformácia (FT) reverzibilná, avšak FT nám dáva informácie len o frekvenčnej oblasti. Niekedy je však potrebné poznať v akom časovom rozmedzí sa v signáli daná frekvencia vyskytla. Odpoveď na to nám môže dát VT. Táto informácia však nie je dôležitá u stacionárnych signálov, u ktorých sa frekvencia nemení v čase. Ak teda potrebujeme mať časovú lokalizáciu frekvenčných zložiek, použijeme transformáciu, ktorá poskytne časovo-frekvenčnú lokalizáciu signálu. Touto transformáciou môže byť VT alebo STFT (Short Time FT). VT sa rozvinula ako alternatíva STFT. Pri časovo-frekvenčnej lokalizácii signálu platí Heisenbergův princíp neurčitosti. Ten tvrdí, že nie je možné presne vedieť aká frekvencia existuje v danom časovom okamihu, teda aké frekvenčné zložky sa vyskytujú v danom časovom intervale. VT na rozdiel od STFT umožňuje premenlivé rozkladanie v čase. Vyššie frekvencie sa lepšie analyzujú v čase a nižšie frekvencie sa lepšie analyzujú vo frekvenčnej oblasti. To znamená, že pre vyššie frekvencie môže byť menší časový interval a naopak. U rozlišovacieho problému sa teda rieši otázka vhodnej voľby časového intervalu, ktorým sa ovplyvní frekvenčné rozlíšenie signálu.[5]

4.1.2 Ako funguje

Diskrétna vlnková transformácia (DWT) poskytuje dostatočné informácie pre analýzu a syntézu originálneho signálu. Hlavná myšlienka je časovo-frekvenčná reprezentácia digitálneho signálu, ktorá sa získa použitím digitálnych filtrov. Rozdielne filtre dekomponujú analyzovaný signál na odlišnej frekvencii. To znamená, že sa vstupný signál nechá prejsť radou filtrov typu horná priepusť k analýze vyšších frekvencií a radou filtrov typu dolná priepusť k analýze nižších frekvencií. Signál sa delí na aproximáciu (reprezentovaná nižšími frekvenciami spracovaného signálu) a detailnejšie informácie (reprezentovaná vyššími frekvenciami spracovaného signálu). Rozlíšenie

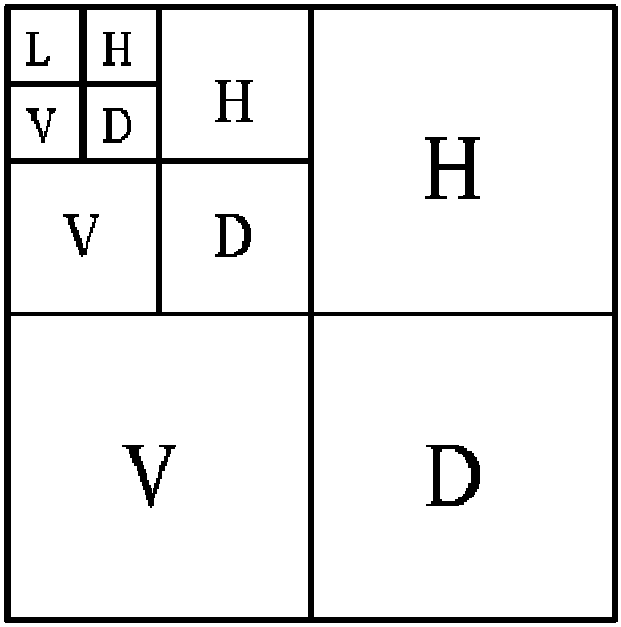
signálu, ktoré je dané množstvom detailných informácií v signálu, je zmenený filtračnými operáciami. Potom nasleduje podvzorkovanie (downsampling). Podvzorkovaním sa odstráni časť vzorku v signálu. Pri syntéze signálu sa k signálu pridávajú detailné informácie a postupuje sa opačne ako pri analýze. Teda signál sa najprv prevzorkuje (upsampling) a potom sa po priechode cez filtre, odlišnými od filtrov použitých pri analýze pridajú detailné informácie. Prevzorkovaním sa pridajú nové vzorky do signálu buď nulové hodnoty alebo interpolačné hodnoty medzi dvoma nasledujúcimi vzorkami a umiestni sa medzi pôvodné vzorky tak, aby sa nachádzali na nepárnych pozíciách. DWT koeficienty sú obvykle vzorkované na dyadickej mriežke (mocniny dvojky).

Pri 2D obrázkoch môžeme využiť to, že waveletová transformácia je oddeliteľná, teda môžeme spracovať najprv stĺpce a potom riadky. Týmto spôsobom nedostaneme dve skupiny koeficientov, ale štyri – aproximačné koeficienty CA, horizontálne CH, vertikálne CV a diagonálne CD (Obrázok 6,7).



Obrázok 6: Demonštrácia vlnkovej transformácie

(zdroj: literatúra 6)



Obrázok 7: Rozčlenenie obrázku na aproximačné koeficienty

(zdroj: literatúra 6)

Po tomto kroku môžeme na takto rozložený obraz aplikovať skalárne kvantovanie (zrovnanie a orezanie hodnôt podľa definovanej rozhodovacej úrovne) koeficientov a výsledok uložiť. K rekonštrukcii musíme previesť opak decimácie (doplnenie výsledku nulami, nadvzorkovanie) a potom využijeme analogické rekonštrukčné filtre čím vrátime systém do pôvodného stavu. V praxi sa ale používajú iné metódy ako SPIHT alebo EBCOT. Dôvodom je to, že pri týchto metódach sa prahovanie môže uskutočniť v krokoch teda v každom kroku sme schopný pridávať k obrazu nové detaily. Druhá výhoda je miera kompresie, ktorá aj pri extrémne malej veľkosti dokáže zachovať obraz rozoznateľný.

5 WSQ

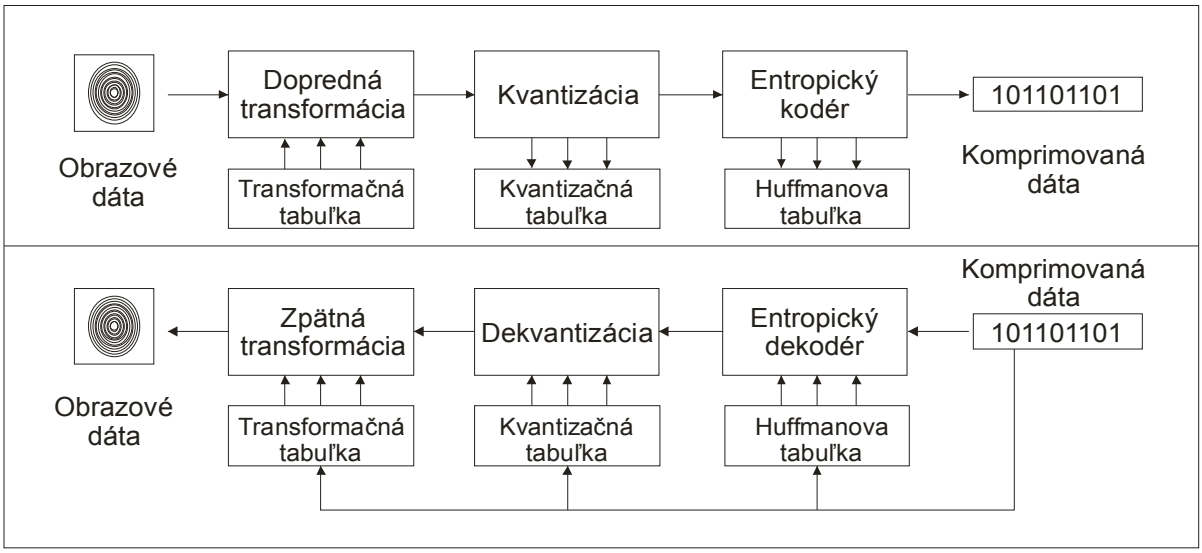
WSQ je vlastne formát vyvinutý pre FBI, a slúži pre ukladanie odtlačkov prstov. Odtlačky prstov sú dosť citlivé informácie, a musí sa s nimi zachádzať opatrne a citlivo. Vlastný formát však nebol potrebný preto, že by chceli tieto informácie zakódovať tak, aby nebol prístupný verejnosti. Dôvod bol iný. Predstavme si, že FBI uchováva odtlačky prstov 114 miliónov kriminálnikov a toto číslo sa deň čo deň zvyšuje. Ak berieme do úvahy, že jeden odtlačok prsta bez kompresie je zhruba 600 kilobajtov dát pri 8 bitovej farebnej šírke, tak celá databáza FBI by dosahovala rozmerov 1140 terabyteov. Tieto dáta sú potrebné v iných pobočkách a miestach a bolo by nemožné ich prenášať. Takže sa pristúpilo k vyvinutiu formátu, ktorý dnes poznáme pod názvom WSQ. Tento formát nie je však zahrnutý v klasických prehliadačoch obrázkov a je dosť obtiažne zo zakódovaného obrázku dostať informácie bez špecializovaného programu.

5.1 Porovnanie s JPEG2000

Formát WSQ je v porovnaní s formátom JPEG2000 o 20 rokov starší. WSQ je špecializovaný formát pre odtlačkov prstov, kým JPEG2000 má obecné použitie ako pri fotografiách, tak pri obrazcoch generovaných počítačom. WSQ dokáže pracovať len s čiernobielymi dátami, kým JPEG2000 nie je viazaný ani formátom RGB. To znamená, že si užívateľ môže vybrať aký typ farebného modelu bude použitý. V oboch formátoch však majú spoločné rysy kompresie, znázornených na obrázku 5. JPEG2000 je však oveľa náročnejší algoritmus, nakoľko sa pri modeli RGB celý proces opakuje 3krát. Priebeh kvantovacieho a a entropického kódovania prebieha pri oboch formátoch podobne.

5.2 Algoritmus

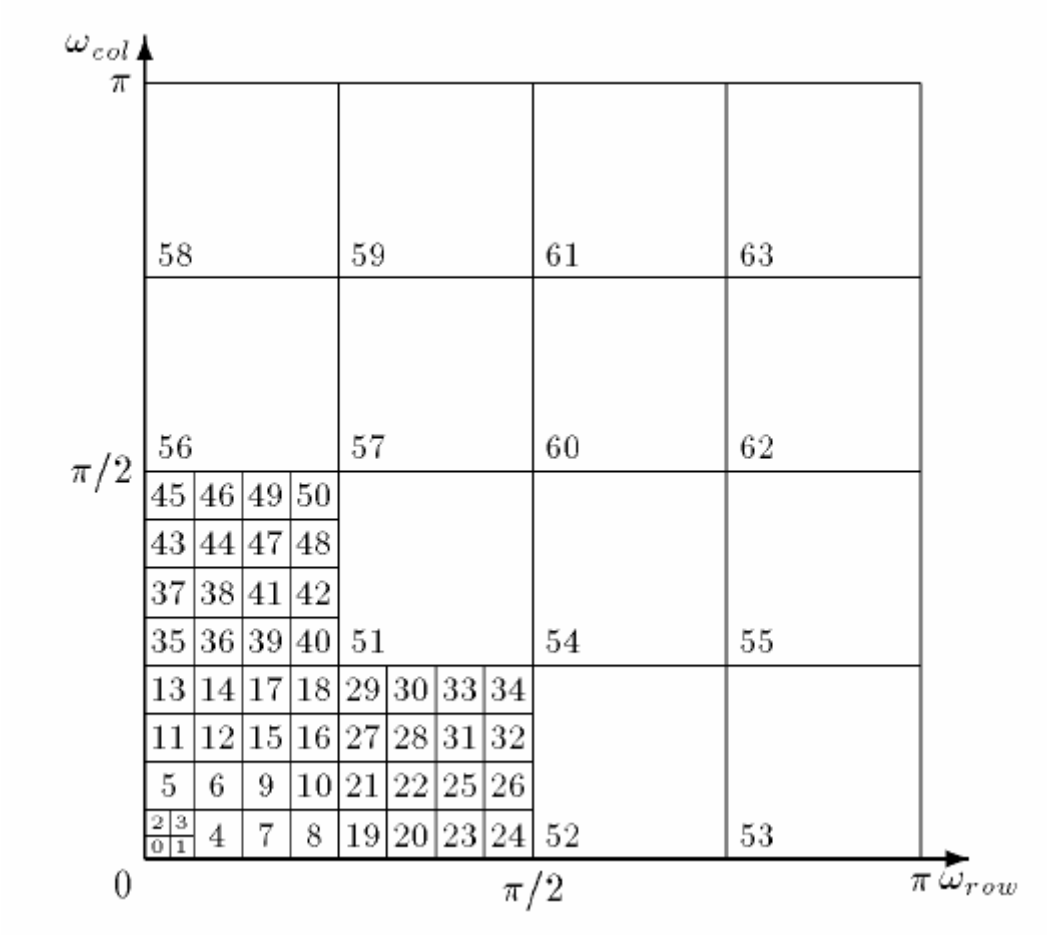
Vývojom tohto algoritmu sa zaoberali vedci v Národnom Laboratóriu Los Alamos a pod FBI fungujúcej Criminal Justice Information Services Division. Národná inštitúcia pre štandardy a technológie zaviedla štandard pre digitalizácie odtlačkov prstov a stratovej obrázkovej kompresie.
Algoritmus enkodéra a dekodéra WSQ obrázku tak isto ako pri JPEG2000 má 3 hlavné kroky. Tieto kroky sú znázornené na obrázku 8. Prvou je popredná transformácia, ktorá rozčleňuje obrázok na 64 dekompozičných častí. Výsledkom sú vlnkové koeficienty, na ktoré sa aplikuje kvantovanie, ktorej výsledkom je kvantovacia tabuľka. Tretí krok zahrňuje entropický enkodér, ktorý bity zakódujú podľa daného kódovacieho algoritmu. Celý postup je znázornený na Obrázku 8.



Obrázok 8: Priebeh komprimovania a dekomprimovania obrázku WSQ

Oficiálna špecifikácia je odkazovaná ako waveletové/skalárne kvantovanie (wavelet/scalar quantization) a je použitý akronym WSQ. Testy FBI ukázali, že obrázky komprimované algoritmom WSQ dosahujú pri 20:1 kompresívnom pomere ešte stále prijateľnú kvalitu pre archiváciu obrázkov.

Diskrétna Vlnková Transformácia vo WSQ algoritmu je implementovaná použitím dvojkanálovej lineárnej fázy filtrovej banky. Symetrické rozširovacie techniky sú použité pre aplikáciu filtrov pri hraniciach obrázku a tak je umožnená transformácia obrázkov ľubovoľných rozmerov (nepárnych čísiel). Toto dvojkanálové rozčlenenie je aplikované pre stĺpce a riadky, (Obrázok 6), čo dáva za výsledok štvorkanálové dvojdimenzionálne rozčlenenie. Táto analýza filtrovej banky je postupne aplikovaná niekoľkokrát za sebou ktorá vygeneruje výsledných 64 dekompozičných častí znázornených na obrázku 10. Výber tejto dekompozície bol uskutočnený podľa spektrálnej výkonovej analýzy obrázkov obsahujúce odtlačky prstov a podľa subjektívnych vyhodnotení kvantovaných artefaktov produkované rôznymi frekvenčnými rozčleneniami.



Obrázok 10: Rozloženie na 64 dekompozičných častí

(zdroj: literatúra 6)

64 DWT dekompozičných obrázkov sú kvantované podľa uniformnej skalárnej kvantovej charakteristiky. Výsledné hodnoty kvantovania sú entopicky kódované pomocou Huffamnovho kódovania. Skalárne kvantovanie a Huffmanov kóder je závislý na obrázku. Výsledné komprimované dáta obsahujú tabuľku použitej vlnkovej transformácie, a tabuľky pre skalárne kvantovanie a Huffmanovo kódovanie. WSQ dekodér musí rozoznať reprezentáciu dát a extrahovať tabuľky potrebné pre rekonštrukciu obrázku. Pri reprodukovaní obrazu dekódované, kvantované koeficienty sú spracované inverzným DWT algoritmom.

Štandard špecifikuje niekoľko typov zakódovaní obrázkov a jediný dekodér s určitou všeobecnosťou pre dekódovanie komprimovaných obrázkových údajov produkovaných vhodnými enkodérmi. Ako pri každej stratovej kompresii je dôležité kontrolovať kvantovací proces, aby bola zabezpečená jednotná kvalita rekonštruovaného obrázku. [6]

5.2.1 Optimálna bitová alokácia

Pri diskutovaní tohto problému musíme brať do úvahy pomer deformácie modelu. Toto sa. Väčšinou rieši pomocou Lagrangovho násobiaceho modelu. Funkcia ktorá má minimalizovať optimalizačný problém je založená na štvorcovom rozložení indukovaným kvantovaním. Tento proces naznačuje nasledujúci vzorec:

$$D = \sum_{k=1}^{M} \frac{1}{m_k} \alpha_k^2 \, \sigma_k^2 \, 2^{-2r_k} \tag{1}$$

m_k je faktor ktorým sú jednotlivé deliace pásma podvzorkované, α^2_k a σ^2_k znázorňujú kvantovaciu chybu k-tého deliaceho pásma, r_k udáva prenosovú rýchlosť (bit rate) obrázku ktorá má byť alokovaná pre zakódovanie k-teho deliace pásma a D je rozdiel chyby ktorá je medzi originálnym a rekonštruovaným obrázkom. Ako vidíme najdôležitejší pre nás je r_k, ktorý v sebe nesie náš cieľ, teda dosiahnuť čo najmenšiu bitovú alokáciu, teda najväčšiu úsporu miesta pri prijateľnej strate Musíme teda použiť už spomínaný Lagrangov násobiaci model. Výsledný vzorec má nasledujúci tvar:

$$r_k = \frac{R}{S_j} + \frac{1}{2} \log_2 \frac{\alpha_k^2 \sigma_k^2}{\left[\prod_{i \in K_j} \left(\alpha_i^2 \sigma_i^2\right)^{1/m_i}\right]^{1/S_j}} \tag{2}$$

V tomto vzorci sa nachádza R, čo je celková prenosová rýchlosť celého obrázku a S_j, ktorý je zlomok j-tej iterácie. Táto iterácia sa mení pri každom deliacom pásme. K_j je inicializačná hodnota iterácie pre dané deliace pásmo. Všetky tie hodnoty, kde r_k dosahuje negatívnu hodnotu, tak sa priradí nulová hodnota a vynechá sa pri výslednej alokácii, čím sa dosiahne úspory miesta pri komprivaných dátach.

5.2.2 Návrh kvanizéra

Kvantizér musí byť navrhnutý tak aby pokryl interval ($\mu - \gamma\sigma_k$, $\mu + \gamma\sigma_k$) v k-tom deliacom pásme. Hodnota γ je zaťažujúci faktor, ktorý udáva štandardnú odchýlku dát, ktoré budú kódované Huffmanovým kódovaním. Ak máme tento návrh znázorniť, tak dostávame nasledujúci vzorec:

$$Q_k = \frac{\gamma 2^{1-R/S} \left[\prod_{i \in K} \left(\alpha_i^2 \sigma_i^2\right)^{1/m_i}\right]^{1/2S}}{\alpha_k} \quad . \tag{3}$$

Výsledná hodnota je teda závislou od koeficientov, ktorých význam bol už uvedený v predchádzajúcej časti. Musíme ale brať do úvahy, že tieto koeficienty sú obrázkovo špecifické, teda K a S sú dané deliacim pásmom, kým R je relatívnym parametrom celého obrázku a závisí od rozmerov obrázku teda od počtu pixlov.

5.3 Štruktúra súboru

Pre skonštruovanie dekodéru je nevyhnutné poznať štruktúru súboru. V tejto časti sa teda budeme zaoberať jednotlivými značkami, a dátami vo WSQ súbore. Zloženie súboru určujú takzvané markery teda značkovače. Každý značkovač je špecifický dvojbajtový kód a má nejaký význam. V súbore sa vždy vyskytujú ako big endian. Niektoré značkovače naznačujú začiatok a koniec obrazových dát, iné určujú parametre. Nakoľko celá štruktúra súboru je navrhnutá ako jeden bitový stream musí byť kladený dôraz na dodržovanie týchto značiek. Ak niektorá značka chýba, celý súbor sa stáva nečitateľným, teda neplatným. Za značkovačmi, ktoré určujú parametre nasleduje takzvaný značkovací segment, uzavretý značkovačom, takže orientácia v súbore je pomerne jednoduchá. Začiatočná značka je SOI teda začiatok obrázku (Start Of Image) a konečná EOI teda koniec obrázku (End Of Image). Taktiež sa tu nachádza SOF čo je začiatok rámca (Start Of Frame). Medzi SOI a SOF sú tabuľky Huffmanovho kódovania, kvantovacie a dekompozičné tabuľky. Za značkou SOF sú uložené dáta obrázku. Podobné značkovače nájdeme aj pri formátu JPEG i JPEG2000.

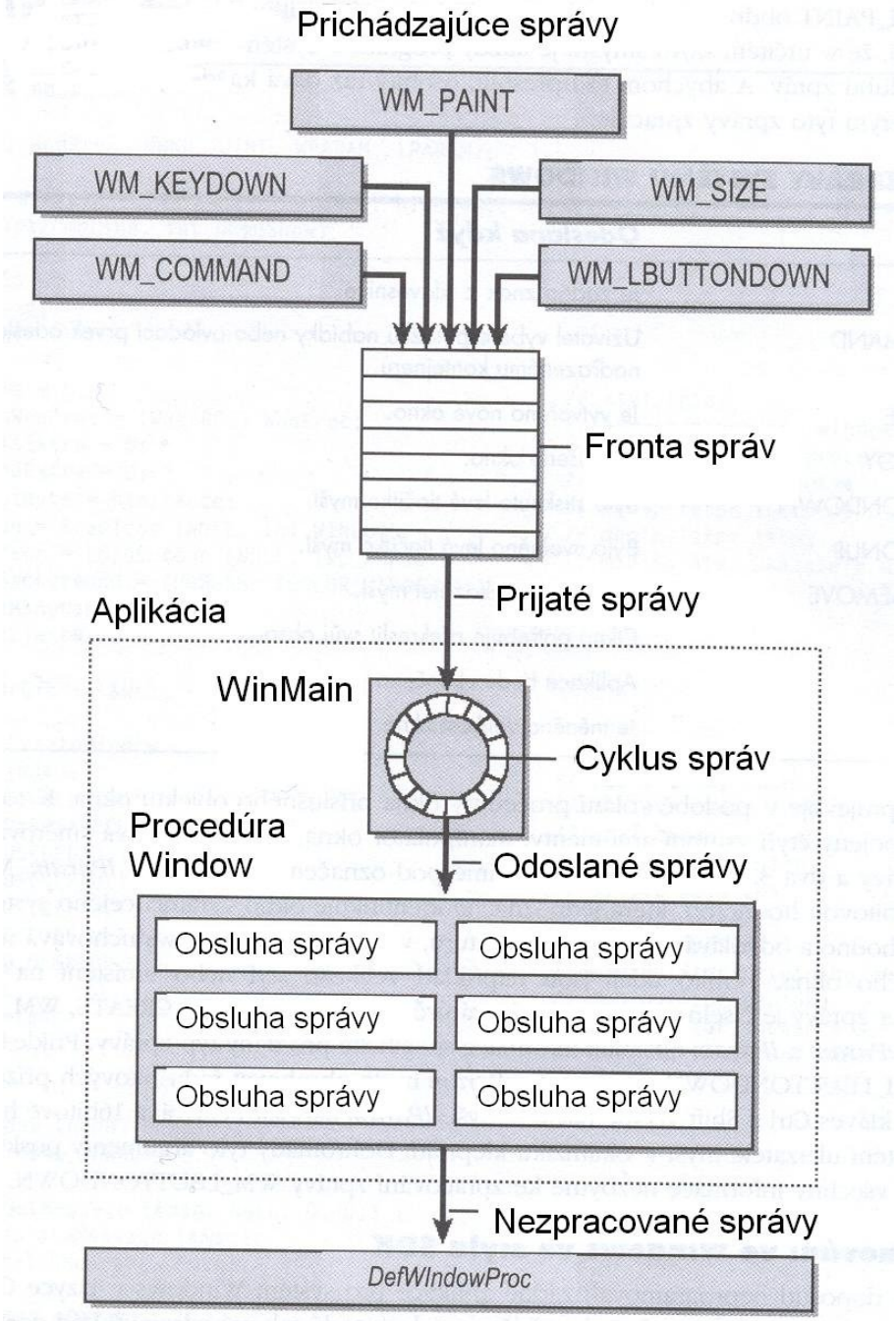
6 Grafické užívateľské rozhranie

Ako každý prehliadač obrázkov musí mať i tento nejaké grafické užívateľské rozhranie. Toto rozhranie umožňuje, aby užívateľ mohol svoju prácu vykonávať bez nutnosti poznať vnútorné funkcie, parametre funkcií, spôsob ich volania, a predávania výsledkov. Nebolo to ale vždy tak. Pred zopár desiatkami rokmi mali hlavné slovo riadkové programy, ktoré sa spúšťali z príkazového riadku. Aj v súčasnosti sa stretávame s takýmito programami, ale pre riešenie tejto úlohy by asi neboli optimálnym riešením. Tento spôsob by sme mohli uplatniť akurát pri hromadnej konverzii, ale to by nám spôsobovalo zbytočné problémy. Teda je nutné vytvoriť grafické rozhranie, s ktorým si poradí každý, kto sa aspoň trochu vyzná v počítačoch.

Pri výbere programovacieho jazyka som neváhal použiť jeden z najznámejších a najrozšírenejších programovacích jazykov C++. Pri vytváraní programu som taktiež bral do úvahy, že program bude používaný pod platformou Windows, a teda je samozrejmý výber nejakej knihovne, ktorá má v sebe zapuzdrené funkcie pre aplikačné programy, skratkou API (Application Programming Interface), podporované operačným systémom Windows. Existuje sada takýchto knihovní a výber môže byť dosť ťažký. Drvivá väčšina programátorov C++ pre systém Windows si vybrala Microsoft Foundation Class, známu pod akronymom MFC. Táto knihovňa bola vytvorená spoločnosťou, ktorá vytvorila aj samotný operačný systém, a nakoľko zadanie obsahuje len platformu Windows, zdá sa rozumným riešením vybrať si MFC. Je síce už stará, ale stále aktualizovaná, umožňuje pracovať pohodlne a taktiež je dobre zdokumentovaná.

6.1 Programovanie pod Windows

Programy navrhnuté pre prostredie tradičných operačných systémov používajú procedurálne programovanie. To znamená, že na základe vstupu program prebehne, prevádza výpočty až sa dostane na výstup. Toto celé sa deje na základe implementácie. Aké funkcie volá programátor tie sa prevádzajú. Takýto program je prehľadný a ľahký na pochopenie. Operačný systém nemá vôbec žiadny vplyv na beh programu. Programy vo Windows fungujú odlišne. Využívajú udalosťami riadený model (Obrázok 9). To znamená, že reakciou na jednotlivé udalosti je vyslanie správ. Udalosť je stlačenie klávesy, alebo pohyb myšou.



Obrázok 9: Správy vo Windows

(zdroj: literatúra 7)

Každá aplikácia vo Windowse musí mať svoj vstupný bod. Po tomto vstupnom bode môže nasledovať procedúra vytvorenia okna, ktorá má za úlohu vytvoriť viditeľnú časť aplikácie, teda okno. Ako som už spomínal Windows komunikuje cez správy. Tieto správy sú adresované niektorému oknu. Následne sa dostanú do fronty správ okna, kde čakajú na vybavenie. Aplikácia sa však nemusí vysporiadať s každou správou. Väčšina implementácií je navrhnutých len pre spracovanie správ, ktoré sú dôležité pre danú aplikáciu. Existuje niekoľko základných správ, na ktoré by mal každý program napísaný pod Windows reagovať. Takýmito správami je zatvorenie okna,

ukončenie programu, minimalizácia, maximalizácia, ale je zodpovedný za to programátor, aby tieto funkcie boli správne implementované. Niektoré správy môžu zostať nespracované, teda bez reakcie. Pri písaní aplikácie teda musíme brať ohľad na tieto skutočnosti a pri návrhu postupovať tak, aby programová štruktúra bola založená na tom, že nedokážeme predpokladať kedy aká správa príde. [7]

6.2 MFC

MFC je knihovňou jazyka tried C++ poskytovanú spoločnosťou Microsoft k vytvorení objektovo orientovaného obalu okolo rozhraní Windows API. Verzia 6 obsahuje okolo 200 tried. Existujú aj úplne jednoduché triedy, napríklad ktoré nakreslia bod na danú súradnicu ale aj komplikované, ktoré zapuzdrujú funkčnosť celého okna. V programu typu MFC sa len zriedka volajú priamo funkcie API. Väčšinou sa vytvárajú objekty a volajú sa funkcie, ktoré sú členom objektov. Tieto funkcie majú v sebe zapuzdrené základné funkcie API, a tak dostávame ten istý výsledok oveľa jednoduchšou a elegantnejšou metódou.

MFC je taktiež aplikačným rámcom. Nie je len kolekciou tried, ale taktiež napomáha pri definovaní štruktúry samotnej aplikácie. Vysporiada sa s bežnými činnosťami v prospech celej aplikácie. Zoberme si ako príklad CWinApp. Táto trieda reprezentuje samotnú aplikáciu a v MFC zapuzdruje všetky aspekty činnosti programu. Tento rámec podporuje funkciu WinMain ktorá volá členské funkcie objektu a tak zaisťuje plynulý chod programu.

6.2.1 Výhody

Už samotná metodológia objektovo orientovaného programovania má v sebe veľa výhod ako napríklad vytvorenie objektu, dedičnosť či zapuzdrenie. Ale bez dobrej knihovne tried, ktorá by slúžila ako základný stavebný kameň aplikácie, by sa programátor asi nezaobišiel bez veľkého množstva zdrojového kódu, ktorý by si musel vytvoriť sám. A tak prichádza MFC. Berme si ako príklad Panel nástrojov, ktorý je možné premiestniť z hornej časti okna do pravej časti, alebo používať ako plávajúci panel nástrojov. MFC má v sebe zabudovanú triedu CToolBar, ktorá všetky tieto funkcie poskytuje, napríklad ak potrebujeme dynamické pole alebo zoznam. Existujú zabudované a používateľné triedy CArray alebo CList. MFC taktiež používa množstvo trikov preto, aby zabudované objekty systému Windows ako okná, dialógové boxy chovali ako objekty C++.

7 Implementácia

Nakoľko sme sa už oboznámili z predchádzajúcich bodov dokumentácie s informáciami potrebnými pre návrh prehliadača, môžeme sa v tejto kapitole venovať priamo praktickému riešeniu problému. Najprv musíme brať ohľad na niekoľko faktov, ako je výzor, praktickosť a efektívnosť riešenia. Hlavnými časťami sú načítanie obrázku a bitový stream, implementácia dekodéru WSQ obrázku, zobrazenie dekódovaného obrázku, a prípadné uloženie obrázku do súboru so zadaným formátom. Taktiež je dôležité vybrať si prostredie pre implementáciu. Existuje niekoľko prostredí, s ktorými je práca komfortná, ktoré majú k dispozícii mnoho pomôcok ale prepracovanú pomôcku MFC a iných windowsových funkcií ponúka len Visual Studio. Takže som si zvolil verziu 2005, a cez svoju robustnosť poskytuje prehľad medzi jednotlivými súbormi a poskytuje dobrý základ pre implementácie vlastnej práce.

7.1 Výzor

Pri aplikácii prvý dojem vytvára jeho výzor. Nakoľko ide o aplikáciu, ktorá má byť špecifická a určená i pre laickú verejnosť, ktorá by sa vôbec nevyznala v rôznych prepínačoch alebo komplikovaných postupoch, je najvhodnejšie vytvárať jednoduchý a prehľadný výzor. Väčšina programov pod Windows používa tento princíp. Stáva sa, že ak program má príliš veľa funkcií, jednotlivé funkcie sa stávajú mätúcimi. S týmto sa však teraz netreba zaoberať, nakoľko program je špecifický, a má za úkol dekódovať obrázky WSQ.

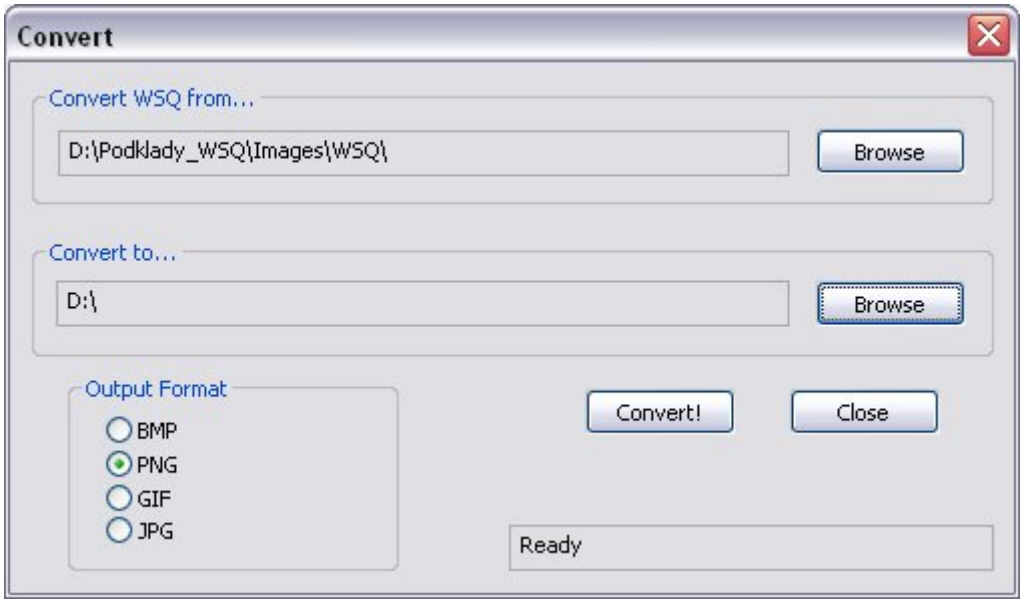
Pre celkový výzor je použité klasické rozloženie komponentov Windows. Aplikácia má menu, štandardný panel nástrojov, okno pre vykreslenie obrázku a stavový riadok (Obrázok 11). V menu a paneli nástrojov sú tie isté funkcie aplikácie. Panel nástrojov je však oveľa jednoduchší a používanie je rýchlejšie, na druhej strane menu je názornejšie a nie je potreba používať myš, môže sa riadiť pomocou vstupov z klávesnice. Pre jednoduchosť ovládania je možné funkcie volať pomocou skratiek. Každý, kto pracoval s nejakým programom pod Windows určite pozná klávesové skratky ako CTRL+S pre uloženie alebo CTRL+O pre otvorenie. Tieto možnosti sú zahrnuté aj v tejto aplikácii.

Obrázok 11: Pohľad na výslednú aplikáciu

Ďalšou možnosťou je vypnutie a zapnutie panelu nástrojov a stavového riadku. Taktiež je možné panel nástrojov premiestniť na ľubovoľné miesto. Stavový riadok poskytuje informácie o mierach obrázku, a rýchly prehľad o tom, čo robí tlačidlo alebo položka v menu ktorá je vybraná.

7.1.1 Dialógové okno Convert

V MFC sme schopný vytvoriť aj vlastné okná, s vlastným rozložením komponentov. Tieto okná sa nazývajú dialógové okná. Práve takéto okno bolo za potrebu pre hromadnú konverziu. Užívateľ musel mať prehľad o tom, s ktorým tlačidlom sa čo nastavuje, aby výsledok bol podľa jeho predstáv. Obrázok 12 znázorňuje toto dialógové okno.



Obrázok 13: Dialógové okno convert

7.2 Dekodér

Dekodér obrázku je jadrom celej aplikácie. Celý projekt naväzuje na túto časť. Na druhej strane je výpočtovo najnáročnejšou časťou, takže implementácia musí byť efektívna. Práca s pamäťou musí byť taktiež dobre prepracovaná. Dekodér nie je implementovaný ako typický C++ kód, ale nakoľko implementácia v jazyku C++ umožňuje spätnú kompatibilitu kódu, táto časť je v jazyku C. K tomuto riešeniu som pristúpil preto, že som mal dobré podklady v jazyku C, ktoré bolo možné pri implementácii použiť. Tieto zdrojové kódy boli súčasťou prehliadača obrázkov WSQ pod unixovou platformou. Obsahovali už implementovaný dekodér, ktorý som použil ako základ môjho dekodéru. Dané kódy sú zmienené v časti literatúra pod číslom 9 [9].

Oboznámili sme sa v kapitole WSQ, so štruktúrou súboru a algoritmom. Tieto informácie nám budú nezbytné pre implementáciu. Hlavná dekódovacia funkcia má názov wsq_decode_mem. Vstupné parametre funkcie musí byť veľkosť predávaných komprimovaných dát a samotné komprimované dáta. Ako výstupné parametre sa vráti pixmap obrázku, šírka, výška a bitová hĺbka.

Proces dekódovania obrázku sa začína inicializáciu tabuliek potrebných pre dekódovanie obrázku. Po inicializácii sa prevádza syntaktická analýza samotných dát. Tu sa skontroluje či je značkovač SOI, na svojom mieste. Ak sa nenájde značkovač, funkcia dekódovania vráti chybové hlásenie.

Po tomto procese sa naplní tabuľka, ktorá bola inicializovaná na začiatku príslušnými dátami, aby mohol proces dekódovania prebehnúť v poriadku. Tieto dáta sa načítajú až po značku SOF. Od tejto značky nasleduje hlavička komprimovaného obrázku. Hlavičku prečítame a následne zistíme aké má obrázok miery a vypočíta sa celkový počet pixlov.

Po tomto kroku sa vytvorí strom dekompozície, naalokuje sa potrebná pracovná pamäť a prevedie sa huffmanovo dekódovanie. Po tomto kroku sa urobí dekvantovanie. Tu ešte stále pracujeme len s podobrázkami takzvanými subbandmi. Následná rekonštrukcia podobrázkov do celku sa vytvorí až v poslednej fáze dekompresie. Po rekonštrukcii je obrázok v prevedení sivej stupnice tak, že jeden pixel je reprezentovaný jedným bajtom. Následne sa dáta odovzdajú ako vrátené hodnoty. Spracovanie týchto dát realizujú nižšie popísané funkcie..

7.3 Funkčnosť

K jednotlivým tlačidlám potrebujeme naimplementovať príslušnú funkčnosť, aby výsledná aplikácia bola použiteľná. Pozrime sa teda bližšie čo sa odohráva v aplikácii ako reakcie na jednotlivé udalosti.

7.3.1 Init

Pod týmto kľúčovým slovom si predstavme inicializáciu samotnej aplikácie. To sa odohráva hneď po spustení aplikácie. Najprv sa zavolá Funkcia CMainWindow, ktorá má za úlohu vytvoriť Samotné okno podľa dopredu definovanej konštrukcie. Táto konštrukcia zahrňuje vytvorenie menu, panelu nástrojov a stavového riadku, dopísať do nich príslušné hodnoty, ktoré sa majú užívateľovi zobraziť. Taktiež v tomto stave sa zaregistrujú správy a sa spustí fronta správ, ktorá má za úlohu spravovať správy prichádzajúce od objektov. Po prebehnutí celej inicializácie je program pripravený pre použitie.

7.3.2 Open

Pri kliknutí na toto tlačidlo sa v aplikácii spustí funkcia OnFileOpen, ktorá volá konštruktér MFC pre OpenFileBox. Následne na to sa otvorí dialógové okno, v ktorom si užívateľ môže vybrať súbor pre otvorenie. Má na výber z dvoch typov obrázkov. Jeden je WSQ obrázok, ktorý chce zobraziť. Popri tom užívateľ dokáže aj otvoriť obrázky klasické, bitmapové, ba až v niekoľkých komprimovaných formátoch ako je napríklad GIF, JPEG či PNG. Podľa tohto výberu, ktorý si vyberie užívateľ sa priebeh programu rozlišuje.

- Ak si vyberie formát WSQ, prečíta sa celý obsah obrázku do dynamicky alokovaného poľa a predá sa funkcie ktorá zabezpečí dekompresiu. Táto funkcia vráti bitovú mapu, ktorá presne určuje výsledné farby na jednotlivých pozíciách. Vrátená hodnota je však čiernobiela, a pre zobrazenie na farebnom monitore potrebuje spraviť konverziu. Táto konverzia sa uskutočňuje po vrátení hodnoty funkcie dekompresie. Následne sa uloží bitová mapa do premennej, ktorú má k dispozícii funkcia reakcie na maľbu, teda OnDraw.
- Druhá možnosť je, že užívateľ chce otvoriť obrázok, ktorí je buď bez kompresie alebo je komprimovaný známou metódou. Na získanie bitovej mapy z takého obrázku nie je treba

vytvárať vlastný algoritmus, stačí použiť zabudovanú funkciu. Pravdaže nemusíme brať ohľad na to s akým typom obrázku máme dočinenia aby daný algoritmus nezlyhal. Po získaní bitovej mapy z takéhoto obrázku sa presne ako pri otváraní WSQ obrázka sa môže funkcia OnDraw zaoberať.

Ako bolo spomenuté, funkcia OnDraw uskutočňuje vykreslenie bitovej mapy do okna, teda na zobrazovaciu jednotku užívateľa. Pri tejto metóde musíme brať ohľad na miery obrázku a miery aktuálneho okna. Ak je obrázok presne taký ako okno, nie je žiaden problém zakresliť ho. Ale čo sa stane ak je obrázok menší ako okno? V tomto prípade sa objaví rolovacia lišta, ktorá má za úlohu zobraziť časť obrázku a následne pridať možnosť prejsť k ukrývajúcej sa časti obrázku. Ak je rolovacia lišta aktívna, teda užívateľ ju ťahá, tak sa okno prekresľuje inou časťou obrázku.

7.3.3 Save

Reakciou na toto tlačidlo je podobne ako pri otváraní volaná funkcia, ktorá otvorí dialógové okno, teraz však určené pre ukladanie obrázku. V tomto dialógovom okne si užívateľ vyberie cestu a názov súboru ktorý sa má uložiť. Môže si vybrať štyri rôzne typy obrázkových formátov a to bezstratový 24 bitový BMP obrázok, GIF, PNG alebo stratový JPG. Podľa tohto výberu sa vyberie formát kompresie a následne sa aktuálne zobrazený obrázok uloží na výsledné miesto na disku. Pri nepodarenom uložení sa generuje chybové hlásenie, ktoré užívateľa oboznámi príčinou vzniknutej chyby.

7.3.4 Convert

Toto tlačidlo slúži pre hromadnú konverziu obrázkov WSQ do niektorého z klasických formátov. Po stlačení tlačidla sa objaví dialógové okno (Obrázok 13). Toto dialógové okno musí umožňovať výber zdrojovej a cieľovej zložky. Po stlačení tlačidla Browse sa otvorí dialógové okno OpenFileDialog, kde je možné si vybrať cestu. Oveľa elegantnejší spôsob by bol, ak by sa otvoril BrowseDialog, ale túto možnosť nemá v sebe zabudované MFC, takže sa uspokojíme s OpenFileDialogom. Po výbere cesty sa táto cesta zapíše do textového poľa, ktorý je priamo vedľa tlačidla. Tento krok je potrebný preto, aby užívateľ dostal potrebnú spätnú väzbu o vybranej ceste. Ak máme vybranú a znázornenú cestu, potrebujeme ešte vybrať formát výsledného obrázku. Converter umožňuje taktiež štyri formáty ukladania a to JPEG, BMP, PNG a GIF. Po výberu formátu sa môže spustiť hromadná konverzia stlačením tlačidla Convert. Ak nie je vybraná niektorá zložka, alebo formát, program to hlási príslušným hlásením.

Samotná konverzia prebieha nasledujúcim spôsobom. Najprv sa otvorí cieľová zložka a vytvoria sa indexy na súbory typu WSQ. Tieto súbory sa postupne konvertujú a ukladajú do cieľovej zložky. Pri konverzii sa zmení len prípona súboru na príponu vybraného formátu, názov zostáva ten istý.

8 Záver

Projekt bol vytvorený ako školný projekt, jeho námetom bola konverzia formátu na známe formáty používané pre ukladanie 2D obrázkov. Projekt predstavoval výzvu, nakoľko sa mal implementovať obtiažny algoritmus dekódovania tohto formátu. Ako bolo znázornené, tento formát predstavuje niekoľkomiliónovú databázu otlačkov prstov, a precízna implementácia bola nevyhnutná najmä pre úsporu miesta. Môj prehliadač obrázkov splňuje kladené podmienky. Dekódovanie obrázku však naďalej zostáva náročným procesom, ktorý naväzuje na hardwarový základ počítača. Toto sa prejaví najvýraznejšie pri hromadnej dekompresii, keď sa počítač musí vysporiadať s niekoľkými desiatkami ba až stovkami obrázkov. Hlavne pri hromadnej kompresii sa musel brať ohľad na efektívnosť riešenia, a odstránenie nepotrebných zdrojov z pamäti. Bez toho by nebolo možné konvertovať len niekoľko obrázkov, nakoľko by sa pamäť rýchlo zaplnila.

Na trhu je zopár identických programov, ktoré umožňujú skoro tie isté funkcie a postupy ako tento projekt. Svoj projekt som konfrontoval s jedným voľne dostupným prehliadačom WSQ obrázkov z dielne cognaxon [8]. Tento prehliadač je verziou 2.7, z toho je možné odvodiť skúsenosť tohto projektového tímu. V porovnaní s mojím prehliadačom má niekoľko funkcií do plusu, ako zoom, kopírovanie do schránky a konverziu do 7 rôznych formátov oproti mojim štyrom. Nezanedbateľné je, že tento program umožňuje aj spätnú konverziu do formátu WSQ, teda má implementovaný aj enkodér. Tento program však neponúka panel nástrojov a v porovnaní s mojim konzumuje o 3MB viac pamäti pri otvorení jedného obrázku. Ak by sme mali vyhlásiť víťaza, musíme uznať že WSQ prehliadač tímu cognaxon ponúka lepšiu alternatívu, avšak pre použitie je postačujúci aj môj prehliadač obrázkov.

Z predchádzajúceho odseku však je možné čerpať námety pre rozšírenie WSQ prehliadača. Asi jedným z najzaujímavejších rozšírení prehliadača by sa mohol stať zoom. Musíme brať do úvahy, že sa jedná o obrázky ktoré v sebe skrývajú informácie o detailoch, a bolo by komfortné tieto informácie vycentrovať a priblížiť na monitore. Možným rozšírením by bola implementácia viacerých obrázkových formátov, čo by prispievalo k univerzálnosti prehliadača. Kopírovanie do schránky je taktiež zaujímavá možnosť ale praktickejšie využitie by si našla aj tlač na papier. Možnosť ukladania obrázkov spätne do formátu WSQ môže byť dnes taktiež dôležitým momentom, nakoľko na trhu sa už objavujú ani nie príliš drahé čítačky odtlačkov prstov. Mal som nedávno možnosť vyskúšať čítačku odtlačkov prstov od spoločnosti Microsoft. Táto čítačka umožňovala zamknutie počítača a následné odomknutie pomocou odtlačku. Nebolo potrebné zadať heslo, stačilo priložiť prst a mohli sme pracovať. Táto čítačka taktiež načítala obrázok a porovnala ho s tým, ktorý mala v databáze. Tento zážitok mi dáva podnet na ďalšie rozšírenie môjho prehliadača, a to komparátor, teda porovnávač odtlačkov. Implementácia tohto riešenia by bola asi časovo náročná, ale využitie by mohlo byť dostatočne zaujímavé v rôznych sférach života. Software takéhoto typu určite existuje

v kriminalistike, ale ako som načrtol, čoraz viac sa nájde priaznivcov pre takéto riešenie aj v civilnej sfére. Zoberme si ako príklad našu fakultu, konkrétne vstup do CVT. Teraz používame vstupový systém riadený kartou. Kartu študent môže stratiť, môže byť ukradnutá. Ale čo je s odtlačkom. Stále ho nosíme so sebou a je taktiež jedinečný. A máme ich hneď niekoľko. A pravdaže potrebujeme aj čítačku a dobrý software pre komparáciu, a keď už máme formát pre kompresiu tak prečo ho nevyužiť pre ukladanie.

Toto všetko je pre tento projekt ešte len víziou budúcnosti. Existujú však už hotové riešenia, firmy ktoré navrhujú takéto riešenia, tak prečo sa nepokúsiť aspoň udržať krok, ak máme dobrý základ pre taký systém v podobe jednej jedinej bakalárskej práce.

Zoznam skratiek

Formáty obrázkov:

WSQ	Wavelet Scalar Quantization
JPEG	Joint Photographic Experts Group
BMP	Bitmap
PNG	Portable Network Graphics
GIF	Graphics Interchange Format
TIFF	Tag Image File Format

Ostatné skratky:

FBI	Federal Bureau of Investigation - Federálny Úrad pre Vyšetrovanie
FT	Furierová Transformácia
RGB	Red Green Blue – farebný model
HSB	Hue, Saturation, Brightness – farebný model
SPIHT	Set Partitioning In Hierarchical Trees - algoritmus
EBCOT	Embedded Bitplane Coding with Optimal Truncation - algoritmus
DWT/DVT	Discrete Wavelet Transformation – Diskrétna vlnková transformácia
DCT	Discrete Cosinus Transformation – Diskrétna kosínusová transformácia
SOI	Start Of Image – začiatok obrázku
SOF	Start Of Frame – začiatok rámcu
EOI	End Of Image – koniec obrázku
GUI	Grapical User Interface – Grafické užívateľské rozhranie
API	Application Programming Interface – rozhranie pre programovanie aplikácie
MFC	Microsoft Foundation Classes – knihovna nad C++
CVT	Centrum Výpočtovej Techniky

Literatura

[1] Súkromné gymnázium v Košiciach: *Kódovanie informácie*. Dokument dostupný na URL: http://www.gymbosak.sk/pages/edu/info/grafika/graficke_formaty.doc, 10.05.2007

[2] Jozef Mak: *Grafické formáty*. Dokument dostupný na URL: http://photoshop.szm.sk/photoshop-manual/graficke-formaty.html, 10.05.2007

[3] JPEG2000 compression example. Dokument dostupný na URL: http://www.yourspot.com.br/design/design_exibicao.php?id=1fad0e0615&acao=comentarios, 10.05.2007

[4] ITEE Innovatoin Expo: *Coprocessing in a JPEG2000 Implementation.* Dokument dostupný na URL: http://innovexpo.itee.uq.edu.au/2001/projects/s369272/, 10.05.2007

[5] R.Polikar: *The Wavelet Tutorial*. Dokument dostupný na URL: http://engineering.rowan.edu/~polikar/wavelet.html, 10.05.2007

[6] Jonathan N. Breadley and Christoper M. Brislawn: *The Wavelet/Scalar Quantization Compression Standard for Digital Fingerprint Images*. PDF dokument, 1999

[7] Jeff Prosise: *Programování ve Windows pomocí MFC*. Computer Press Praha, 2002, ISBN 80-7226-309-9

[8] Cognaxon: *WSQ Viewer*. Dokument dostupný na URL: http://www.cognaxon.com/, 10.05.2007

Zoznam príloh

Príloha 1. Zdrojové texty na CD