Obsah

Kapitola 1

Úvod

Ke zpracování obrazu se využívá mnoho technik. Mezi ně patří například segmentace,geometrické transformace, matematická morfologie, a také zpracování obrazu pomocí diskrétních transformací. Diskrétní transformace slouží k zobrazení dat do jiného prostoru, kde se určité operace provádějí snadněji. Typickým příkladem takovýchto transformací jsou filtrace dat.

Cílem této práce je vybrat diskrétní transformace vhodné pro zpracování digitalizovaného obrazu a otestovat jejich funkci.

1.1 Struktura dokumentu

Kapitola Teoretický úvod rozebírá jednotlivé vybrané diskrétní transformace a zkoumá jejich rychlé implementace. Na závěr této kapitoly jsou popsány oblasti použití diskrétních transformací na zpracování digitalizovaného obrazu.

Kapitola Praktická část popisuje implementaci těchto diskrétních transformací a jejich začlenění jako modulů do MVE. Na závěr této kapitoly je umístěno zhodnocení implementace.

Kapitola Závěr shrnuje výsledky práce.

Tato práce je implementována pro modulární a vizualizační prostředí MVE v rámci projektu MŠMT VS 97 155.

Kapitola 2

Teoretický úvod

2.1 Fourierova transformace

2.1.1 Spojitá Fourierova transformace

Fourierova transformace poskytuje prostředky pro konverzi dat mezi jejich dvěma reprezentacemi v datové (časové) a frekvenční (spektrální) oblasti.

Fourierova transformace je zobecněním Fourierových řad na nekonečný interval a je definována

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{-i2\pi ft} \mathrm{d}t \tag{2.1}$$

Reprezentace dat ve frekvenční oblasti obsahuje nezbytnou informaci k rekonstrukci dat do datové oblasti inverzní transformací

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{i2\pi ft} \mathrm{d}f \tag{2.2}$$

kde

$h(t)$ je spojitá funkce $t \in \mathcal{R}$ (v případě zpracování signálů je funkcí času)

$H(f)$ je spojitá funkce frekvence $f \in \mathcal{R}$ (v případě zpracovávání signálů se udává v hertzích hz)

Při zpracovávání signálů pracujeme často s kruhovou frekvencí $\omega = 2\pi f$ (udává se v radiánech za sekundu).V tom případě je Fourierova transformace definována

$$H(\omega) = \int_{-\infty}^{\infty} h(t) e^{-i\omega t} \mathrm{d}t \qquad \text{inverzní } h(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} H(\omega) e^{i\omega t} \mathrm{d}f$$

Fourieriva transformace je lineární zobrazení neboť platí

$$\mathcal{FT}\{ag(t) + bh(t)\} = aG(f) + bH(f) \tag{2.3}$$

kde $\quad G(f) = \mathcal{FT}\{g(t)\} \qquad H(f) = \mathcal{FT}\{h(t)\}$

Další užitečné teorémy jsou uvedeny v tab 2.1. V tab 2.2 jsou uvedeny vlastnosti frekvenčních reprezentací některých typů dat.

Reprezentace dat ve frekvenci se také nazývá *spektrum* a obvykle se zobrazuje jako amplituda a fáze :

$$|H(f)| = \sqrt{\mathcal{RE}^2\{H(f)\} + \mathcal{IM}^2\{H(f)\}} \qquad \sphericalangle H(f) = \arctan\left(\frac{\mathcal{IM}\{H(f)\}}{\mathcal{RE}\{H(f)\}}\right)$$

Většina informace je soustředěna na začátku spektra. Proto je vhodné f vynášet v logaritmické stupnici a amplitudu zobrazovat jako *zisk* : $20 \log |H(f)| = |H(f)|\, dB$.

závislost	datová oblast	frekvenční oblast
změna měřítka v datové oblasti	$h(at)$	$\frac{1}{\|a\|}H(\frac{f}{a})$
změna měřítka ve frekvenci	$\frac{1}{\|a\|}h(\frac{t}{a})$	$H(af)$
posun v datové oblasti	$h(t-t_0)$	$H(f)e^{-i2\pi ft_0}$
posun ve frekvenci	$h(t)e^{i2\pi f_0 t}$	$H(f-f_0)$
derivace v datové oblasti	$\frac{\mathrm{d}^p h(t)}{\mathrm{d}t^p}$	$(i2\pi f)^p H(f)$
derivace ve frekvenci	$(-i2\pi t)^p h(t)$	$\frac{\mathrm{d}^p H(f)}{\mathrm{d}f^p}$
integrace v datové oblasti	$\int_{-\infty}^{t} h(t)\mathrm{d}t$	$\frac{H(f)}{i2\pi f} + H(0)\delta(f)$
integrace ve frekvenční oblasti	$\frac{h(t)}{-i2\pi t} + h(0)\delta(t)$	$\int_{t}^{\infty} H(f)\mathrm{d}f$
komplení sdružení	$h^*(t)$	$H^*(-f)$

Tabulka 2.1: Vybrané vlastnosti Fourierovy transformace

$h(t)$	$\mathcal{FT}\{h(t)\} = H(f)$
reálná	$H(-f) = H^*(f)$
imaginární	$H(-f) = -H^*(f)$
sudá	sudá
lichá	lichá
reálná a sudá	reálná a sudá
reálná a lichá	imaginární a lichá
imaginární a sudá	imaginární a sudá
imaginární a lichá	reálná a lichá

Tabulka 2.2: Vlastnosti frekvenčních reprezentací některých typů dat

Spojitá konvoluce

Uvažujeme-li, že funkce $Y(f) = \mathcal{FT}\{y(t)\}$ je součinem funkcí $H(f) = \mathcal{FT}\{h(t)\}$ a $G(f) = \mathcal{FT}\{g(t)\}$

$$Y(f) = H(f)G(f)$$

pak $y(t)$ je **konvolucí** funkcí $h(t)$ a $g(t)$

$$y(t) = \mathcal{FT}^{-1}\{Y(f)\} = h(t) * g(t) = \int_{-\infty}^{\infty} h(t)g(t-\tau)d\tau \tag{2.4}$$

Konvoluce je komutativní, asociativní, distributivní a obousměrná [1].

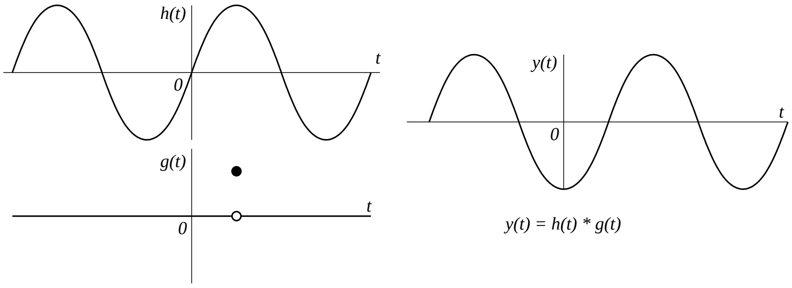
Při zpracovávání signálů se často užívá termín ”konvoluce filtruje”, protože konvoluce v časové oblasti (2.4) znamená odezvu systému (filtru) na vstupní data, tedy filtraci. Konvoluce ve frekvenční oblasti znamená vážení dat nějakou funkcí obvykle takzvaným ”okénkem”.

Vzorkování dat

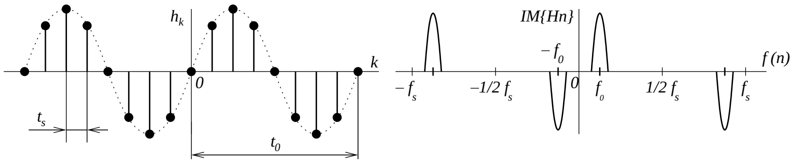
Vzorkování lze chápat jako systém; navzorkovaná data jsou konvolucí spojitých dat a vzorkovací funkce. Fourierova transformace vzorkovací funkce (což je posloupnost jednotkových pulsů vzdálených od sebe t_s) je opět posloupnost jednotkových pulsů vzdálených od sebe $f_s = \frac{1}{t_s}$. Z (2.4) je tedy patrné že frekvenční reprezentace dat bude periodická s periodou f_s. Intervalu $-\frac{fs}{2} \leq f \leq \frac{fs}{2}$ se říká *základní interval* a veškerá informace kterou potřebujeme k rekonstrukci navzorkovaných dat je obsažena v tomto intervalu.

Maximální frekvence, kterou mohou data obsahovat je *Nyquistova frekvence* $f_c = \frac{f_s}{2}$. Pokud se v datech vyskytnou frekvence, které jsou mimo základní interval, pak při jejich vzorkování dojde

[1] tj. $H(f) * G(f) = \int_{-\infty}^{\infty} H(f)G(f-f_0)df_0$



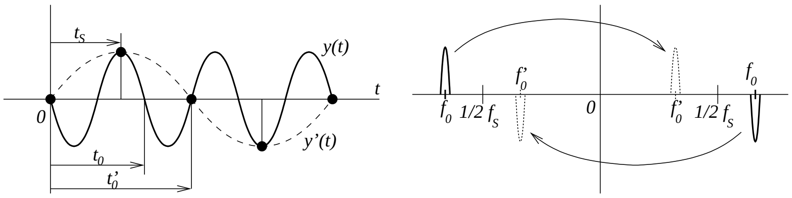
Obrázek 2.1: Příklad spojité konvoluce.



Obrázek 2.2: Vzorkování - periodicita spektra navzorkovaného signálu.

k překrývání spektra (aliasing). Frekvence, které jsou mimo základní interval, se totiž promítnou do základního intervalu a dojde tudíž k znehodnocení spektra.

Překrývání spektra se vyhneme začleněním vhodného antialising filtru, který maximálně po-



Obrázek 2.3: Vzorkování - efekt aliasing.

tlačí frekvence vyšší nežli f_c a minimálně ovlivní frekvence nižší než f_c.

2.1.2 Diskrétní Fourierova transformace

Provedeme estimaci Fourierovi transformace na konečném počtu navzorkovaných dat tedy na vektoru konečné délky.

Předpokládejme N rovnoměrně navzorkovaných dat

$$h_k = h(t_k) \qquad t_k = kt_s \qquad k = 0, 1, \ldots, N-1$$

,kde t_s je perioda vzorkování a N je sudé. Je zřejmé, že můžeme vypočítat pouze N nezávislých hodnot. Omezíme se tedy na základní interval $-\frac{fs}{2} \leq f \leq \frac{fs}{2}$ a budeme hledat estimaci pouze na diskrétních hodnotách

$$f_n = \frac{n}{Nt_s} \qquad \text{kde} \quad n = \frac{-N}{2}, \ldots, \frac{N}{2}$$

Extrémní hodnoty n korespondují přesně se základním intervalem. Hodnoty v $-\frac{N}{2}$ a $\frac{N}{2}$ jsou závislé (ve skutečnosti totožné).

Za těchto předpokladů můžeme aproximovat integrál Fourierovi transformace diskrétní sumou.

$$H(f_n) = \int_{-\infty}^{\infty} h(t)e^{-i2\pi f_n t}\mathrm{d}t \approx \sum_{k=0}^{N-1} h_k e^{-i2\pi f_n t_k} t_s = t_s \sum_{k=0}^{N-1} h_k e^{\frac{-i2\pi kn}{N}}$$

Diskrétní Fourierova transformace (dále jen DFT) je tedy definována

$$H_n = \sum_{k=0}^{N-1} h_k e^{\frac{-i2\pi kn}{N}} \tag{2.5}$$

DFT je úplnou reprezentací vzorkovanýh dat.

Závislost mezi DFT vzorkovaných dat a Fourierovou transformací jejich spojitého originálu může být zapsána jako $H(f_n) \approx t_s H_n$.

Díky periodicitě DFT způsobené vzorkováním je

$$H_{-n} = H_{N-n}$$

Je vhodné volit $n = 0, \ldots, N-1$ tedy jednu kompletní periodu. Pak n a k se pohybují ve stejném intervalu.

Za těchto podmínek

$$\begin{array}{rlllrll} & f & = 0 & \text{odpovídá} & & n & = 0 \\ 0 \le & f & \le f_c & \text{odpovídá} & 1 \le & n & \le \frac{N}{2} - 1 \\ & f & = \pm f_c & \text{odpovídá} & & n & = \frac{N}{2} \\ -f_c \le & f & \le 0 & \text{odpovídá} & \frac{N}{2} + 1 \le & n & \le N - 1 \end{array}$$

Data h jsou zapsána jako kombinace N bázových vektorů. Tyto vektory jsou konstruovány z mocnin komplexního čísla $w = e^{-i2\pi}N$:

$$v_k = (1, w^k, w^{2k}, \ldots, w^(N-1)k), k = 0, 1, \ldots, N-1.$$

Vektory v_k jsou tedy sloupce Fourierovi matice $F = F_N$. Tyto sloupce jsou ortogonální. Tudíž inverze F (inverzní transformace) je její hermitovská transpozice, dělená $\|v_k\|^2 = N$.

Proto inverzní DFT lze zapsat jako

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{\frac{i2\pi kn}{N}} \tag{2.6}$$

Z rovnice (2.6) plyne, že inverzní DFT produkuje vektor nekonečné délky s periodou N.

Všechny teorémy uvedeny v tab. (2.1) platí, pokud integrály přejdou v sumy, derivace v diference a spojité funkce se nahradí vektorem diskrétních hodnot. Též vlastnosti spojité Fourierovi transformace uvedené tab. (2.2) platí pokud funkce přejdou na vektory.

Diskrétní konvoluce

Diskrétní konvoluce má stejné vlastnosti jako konvoluce spojitá. Pokud vyjdeme ze stejných předpokladů jako v kapitole 2.1.1, s tím rozdílem že se nejedná o spojitá, ale o diskrétní data je diskrétní konvoluce definována :

$$y_k = \mathcal{DFT}^{-1}\{Y_k\} = h_k * g_k = \sum_{m=-\infty}^{\infty} h_m g_{k-m} \tag{2.7}$$

V konvoluci ve frekvenční oblasti se též projevuje se "kruhová" periodicita. (viz. kap 2.1.1). Odtud o konvoluci spekter diskrétních dat mluvíme jako o "kruhové konvoluci".

V případě diskrétní konvoluce se tedy doporučuje rozšířit oboje data na délku rovnající se součtu délek obou dat (rozšíření je nutné vyplnit nulami), jelikož by docházelo k překrývání na krajích konvolvovaných dat.

2.1.3 Rychlá diskrétní Fourierova transformace

DFT je výpočetně velmi náročné z hlediska počtu komplexních násobení a výpočtů $e^{\frac{\pm i2\pi kn}{N}}$. Výpočetní náročnost DFT je $\mathcal{O}(N^2)$. Rychlá diskrétní Fourierova transformace (dále jen FFT) provádí výpočet DFT s výpočetní náročností $\mathcal{O}(\frac{N}{2}log_2N)$.

Předpokládáme, že jak data tak i jejich frekvenční obraz jsou komplexní. Dále uvažujeme, že data jsou délky 2^l, kde l je libovolné celé kladné číslo. V případě, že tomu tak není doplníme za data nuly až do nejbližší vyšší mocniny dvou.

Hlavní myšlenkou FFT je rozdělit data na dvě části, spočítat pro ně DFT a dovypočítat z těchto dvou transformací transformaci celého intervalu. Výpočetní náročnost by pak byla $2(\frac{N}{2})^2 = \frac{N^2}{2} \leq N^2$. Totéž provádíme i na takto rozdělených intervalech. Dojde tedy k postupnému hierarchickému rozložení celého výpočtu, vždy na dvě části. Proto je třeba, aby byla velikost dat mocninou dvou (odtud označení radix-2 FFT). Pro data velikosti N provedeme log_2N takovýchto rozdělení.Tento mechanismus se nazývá *decimace*, zde jde o decimaci v čase. Decimaci lze provést i ve frekvenci (DIF). FFT s decimací v čase se také nazývá Cooley-Tukey FFT a FFT s decimací ve frekvenci se označuje jako Sande-Tukey FFT dle jejich autorů.

Decimace v čase - DIT

Jak je uvedeno výše je třeba rozdělit data na dvě části. Rozdělení provedeme podle indexu na sudá a lichá data. Vyjdeme z rovnice (2.5)

$$H_n = \sum_{k=0}^{\frac{N}{2}-1} h_{2k} e^{\frac{-i2\pi 2nk}{\frac{N}{2}}} + e^{\frac{-i2\pi n}{N}} \sum_{k=0}^{\frac{N}{2}-1} h_{2k+1} e^{\frac{-i2\pi 2nk}{\frac{N}{2}}} = \underbrace{H_n^e + e^{\frac{-i2\pi n}{N}} H_n^o}_{\text{Danielson Lancsos Lemma}} \qquad (2.8)$$

Obě dílčí transformace mají bázové funkce periodické v polovičním intervalu, protože $e^{\frac{-i2\pi nk}{\frac{N}{2}}}$ je $k = \frac{N}{2}$ periodické. Díky tomu lze H_n vypočítat i pro druhou polovinu rozsahu od $\frac{N}{2}$ do N bez dalších složitých výpočtů.

Zaveďme subsituci $n = n' + \frac{N}{2}$

$$H_n = H_{n'+\frac{N}{2}} = H_{n'}^e + e^{\frac{-i2\pi(n'+\frac{N}{2})}{N}} H_{n'}^o = H_{n'}^e - e^{\frac{-i2\pi n'}{N}} H_{n'}^o \qquad (2.9)$$

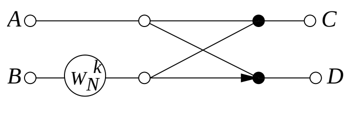
Pokud tedy zaveme $W_N = e^{\frac{-i2\pi}{N}}$ pak základní výpočetní jednotka FFT s decimací v čase je

$$\begin{aligned} C &= A + BW_N^n \\ D &= A - BW_N^n \end{aligned} \qquad (2.10)$$

Výpočet byl tedy rozdělen na výpočet dvou $\frac{N}{2}$ bodových DFT, $\frac{N}{2}$ komplexních násobení a N komplexních součtů. Tento mechanismus postupného dělení aplikujeme tak dlouho dokud nedostaneme DFT na vektoru délky jedna; DFT délky jedna je $H_0 = h_0$.

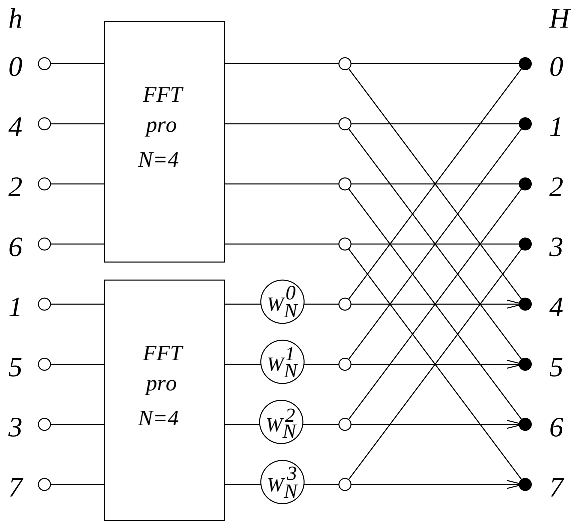
V každém kroku jde vždy o jedno komplexní násobení a dva součty dvou prvků. Tento výpočet provádí základní výpočetní jednotka FFT nazývaná *butterfly*. Buttefly kombinuje dva vstupy do dvou výstupů.

Této třídě FFT se říká *s výpočtem v místě*[2].Obvykle je algoritmus velmi jednoduchý a zároveň



Obrázek 2.4: Výpočetní jednotka FFT s decimací v čase (DIT).

efektivní, ale předpokládá jinak seřazená data na vstupu. Seřazení dat na vstupu FFT s decimací v čase se označuje jako *bitově reverzní*[3].



Obrázek 2.5: Výpočetní graf FFT s decimací v čase pro $N = 8$.

Decimace ve frekvenci - DIF

Decimaci lze provést i opačným způsobem, ve frekvenci. Výpočetní náročnost takto získaného algoritmu je stejná jako u decimace v čase.

Ze vzorce DFT (2.5) vybereme sudé a liché frekvence:

sudé frekvence : $H_{2k} = \sum_{n=0}^{\frac{N}{2}-1}(h_n + h_{n+\frac{N}{2}})e^{\frac{-i2\pi 2k}{N}}$

liché frekvence : $H_{2k+1} = \sum_{n=0}^{\frac{N}{2}-1}(h_n - h_{n+\frac{N}{2}})e^{\frac{-i2\pi n}{N}}e^{\frac{-i2\pi 2k}{N}}$

Je tedy vidět, že výpočet FFT s decimací ve frekvenci je velmi podobný FFT se decimací v čase, s tím rozdílem, že hierarchické dělení probíhá opačným směrem. Výpočetní graf FFT s decimací ve frekvenci by se tedy zdál otočený oproti FFT s decimací v čase (2.5).

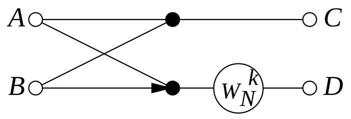
Výpočetní jednotka FFT s decimací ve frekvenci je tedy:

$$\begin{aligned} C &= A + B \\ D &= (A - B)W^k \end{aligned} \tag{2.11}$$

Opět se jedná o FFT s výpočetem v místě. Oproti FFT s decimací v čase nepotřebuje bitově reverzní seřazení vstupů, ale je třeba bitově reverzně seřadit výstupy. U FFT s decimací ve frekvenci lze bitový reverz ve speciálních případech vynechat (např. konvoluce, kde je možné provést konvoluci na spřeházených výstupech), nicméně bitový reverz představuje velmi malou část samotného výpočtu FFT.

[2]což znamená že celý výpočet nepotřebuje zvláštní datový prostor, tedy že výstup přepíše vstup

[3]tj. index reprezenetující umístění ve vstupu je převeden do binární soustavy, tam je "přečten" pozpátku aby se získalo nové umístění pro tento vstup



Obrázek 2.6: Výpočetní jednotka FFT s decimací ve frekvenci (DIF).

Ostatní algoritmy

Jednou z možností jak dále urychlit FFT je například nerozdělovat FFT až na velikosti jedna, ale raději na nějaké vhodné N například $N = 4$ nebo $N = 8$. Tyto malé transformace jsou pak provedeny ve velmi optimalizovaných sekcích algoritmu, které jsou založeny na existenci speciálních symetrií v diskrétní Fourierově transformaci pro takto malá data.

Existují též algoritmy pro data velikosti $N \neq 2^l$. Pracují na podobném principu jako Danielson Lanczos Lemma, tj. rozdělit problém na menší problémy, nikoliv však na násobky dvou, ale na jakékoliv dělitele N. Ovšem tyto metody pracují velmi špatně na datech, jejichž velikost má velmi velké dělitele. Pokud je N prvočíslo tyto algoritmy selhávají a provádějí obyčejné $\mathcal{O}(N^2)$ DFT.

Jedinou zajímavější metodou z této třídy FFT je Winogradova FFT. Winograd odvodil velmi optimalizované algoritmy pro malé N, například pro $N = 2, 3, 4, 5, 7, 8, 11, 13, 16$. Winogradova metoda provádí přeuspořádání před i po hierarchickém výpočtu, a umožňuje signifikantní zvýšení rychlosti (až dvojnásobné) oproti radix-2 FFT. Toto zvýšení rychlosti je ovšem vyváženo značně složitým indexováním dat a také faktem, že Winogradova metoda nemůže být provedena v místě.

Rychlá inverzní diskrétní Fourierova transformace

Stejný algoritmus, který provádí FFT, lze použít i na výpočet inverzní FFT, protože obě rovnice na výpočet DFT a inverzní DFT jsou až na znaménko u mocniny e a na dělení u inverzního DFT stejné.

Struktura obou algoritmů může být zachována jen W^x nahradíme W^{-x}. Po hierarchickém výpočtu butterflies je ještě nutné celá data vydělit N jak vyplývá z definice inverzní DFT (2.6).

Rychlá diskrétní Fourierova transformace reálných dat

Velmi zajímavou vlastností frekvenční reprezentace čistě reálných a čistě imaginárních dat je symetrie okolo nuly (viz. tab. 2.2). Díky této symetrii a linearitě Fourierovi transformace (2.3) je možné z frekvenční reprezentace komplexních dat vypočítat frekvenční reprezentaci jen reálné části dat a frekvenční reprezentaci jen imaginární části dat.

Mejmě data $h_k = f_k + g_k$ $k = 0, \ldots, N-1$ kde f_k jsou reálná data a g_n jsou imaginární data. Pak díky linearitě Fourierovi transformace (FFT) je $H_n = F_n + G_n$ kde $H = \mathcal{DFT}\{h\}$ $F = \mathcal{DFT}\{f\}, G_n = \mathcal{DFT}\{g\}$
Z tab. (2.2) platí:

$F_n = R_n^e + iI_n^o$,kde F_n je frekvenční reprezentace reálných dat

$G_n = R_n^o + iI_n^e$,kde G_n je frekvenční reprezentace imaginárních dat

Za předpokladu, že $H_n = R_n + iI_n$ platí

$$\begin{aligned} R_n &= R_n^e + R_n^o \\ I_n &= I_n^e + I_n^o \end{aligned}$$

kde horní index ^{e} znamená sudou část dat (even) a horní index ^{o} znamená lichou část dat (odd).

Jednoduchými úpravami zjistíme že

$$\begin{array}{rclrcl} R_n^e & = & \frac{R_n+R_{-k}}{2} & R_n^o & = & \frac{R_n-R_{-k}}{2} \\ I_n^e & = & \frac{I_n+I_{-k}}{2} & I_n^o & = & \frac{I_n-I_{-k}}{2} \end{array}$$

Získali jsme obě spektra za cenu jednoho výpočtu FFT. Z této myšlenky lze čerpat i pro urychlení FFT reálných dat.

Mějme reálná data velikosti mocniny dvou jako v případě komplexního radix-2 FFT. Tato data rozdělíme na sudá a lichá, sudou část vložíme do reálné složky pomocných dat a lichou část do imaginární složky pomocných dat. Je zřejmé, že pomocná data budou poloviční délky. Aplikujeme-li na tato pomocná data komplexní radix-2 FFT a získáme z těchto dat frekvenční reprezentace reálných (sudých ve smyslu původních dat) dat a imaginárních (lichých) dat, pak zbývá jen provést poslední krok FFT s decimací v čase.

Rozdělme tedy reálná data $h_k, k = 0, \ldots, N-1$ na sudá $h_k^e = h_{2k}$ a lichá $h_k^o = h_{2k+1}$ $k = 0, \ldots, \frac{N}{2} - 1$. Sestavíme pomocná data $g_k = h_k^e + i h_k^o$. Díky linearitě Fourierovi transformace je opět $G_n = H_n^e + i H_n^o$, kde $G = \mathcal{DFT}\{g\}$, $H^e = \mathcal{DFT}\{H^e\}$, $H^o = \mathcal{DFT}\{H^o\}$. Zbývá pouze vypočítat obě dílčí transformace a provést výpočet butterfly.

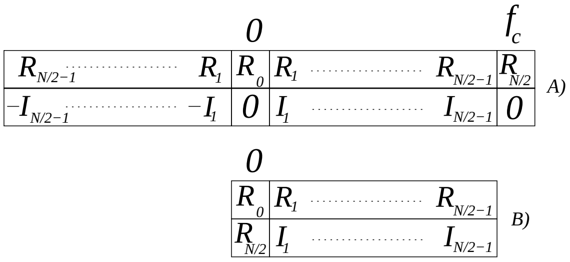
$$H_n = \frac{1}{2}(G_n + G^*_{\frac{N}{2}-n}) - \frac{i}{2}(G_n - G^*_{\frac{N}{2}-n})e^{\frac{-i2\pi n}{N}} \tag{2.12}$$

$$H_{\frac{N}{2}-n} = \frac{1}{2}(G^*_n + G_{\frac{N}{2}-n}) - \frac{i}{2}(G^*_n - G_{\frac{N}{2}-n})e^{\frac{i2\pi n}{N}} \tag{2.13}$$

Problém na okraji vyřešíme periodicitou : $G_{\frac{N}{2}} = G_0$.

Tímto způsobem lze uspořit téměr polovinu času a datového prostoru. Pro úplnou reprezentaci ve frekvenční oblasti je potřeba $\frac{N}{2} + 1$ hodnot, neboť DFT reálných dat má $\frac{N}{2} + 1$ nezávislých hodnot v intervalu $0 \leq k \leq \frac{N}{2}$. Na pozici $\frac{N}{2}$ je totiž Nyquistova frekvence kterou nesmíme vynechat. V jednodimenzionálním případě je vhodné hodnotu na této frekvenci ukládat do imaginární složky H_0.

Inverzní reálnou FFT provedeme reverzí kroků které vedly k reálné FFT. To znamená, že



Obrázek 2.7: **Úsporné uložení spektra reálných dat**: a) DFT reálných dat b) úsporné uložení DFT reálných dat v paměti

zkonstruujeme opět spektrum G :

$$\begin{array}{rcl} H_n^e & = & \frac{1}{2}(H_n + H^*_{\frac{N}{2}-n}) \\ H_n^o & = & \frac{1}{2}e^{\frac{i2\pi n}{N}}(H_n - H^*_{\frac{N}{2}-n}) \\ G_n & = & H_n^e + i H_n^o \end{array}$$

Na spektrum G aplikujeme inverzní komplexní FFT, a dostaneme data g. Tato data je třeba dále poskládat tak, aby reálná část g tvořila sudé vzorky h a imaginární část g tvořila liché vzorky h.

2.1.4 Vícedimenzionální Fourierova transformace

Fourierova transformace spojitých dat dimenze M a její inverze jsou definovány

$$H(f_1,\dots,f_M)=\int\cdots\int_{-\infty}^{\infty}h(t_1,\dots,t_M)e^{-i2\pi f_1t_1}\dots e^{-i2\pi f_Mt_M}\mathrm{d}t_1\dots\mathrm{d}t_M \tag{2.14}$$

$$h(t_1,\dots,t_M)=\int\cdots\int_{-\infty}^{\infty}H(f_1,\dots,f_M)e^{i2\pi f_1t_1}\dots e^{i2\pi f_Mt_M}\mathrm{d}f_1\dots\mathrm{d}f_M \tag{2.15}$$

Pak je zřejmé, že DFT dimenze M a její inverze budou definovány

$$H_{n_1,\dots,n_M}=\sum_{k_1=0}^{N_1}\dots\sum_{k_M=0}^{N_M}h_{k_1,\dots,k_M}e^{\frac{-i2\pi n_1k_1}{N_1}}\dots e^{\frac{-i2\pi n_Mk_M}{N_M}} \tag{2.16}$$

$$h_{k_1,\dots,k_M}=\sum_{n_1=0}^{N_1}\dots\sum_{n_M=0}^{N_M}H_{n_1,\dots,n_M}e^{\frac{-i2\pi k_1n_1}{N_1}}\dots e^{\frac{-i2\pi k_Mn_M}{N_M}} \tag{2.17}$$

Samozřejmě i pro vícedimenzionální data platí pravidla o správném vzorkování (viz. kapitolu 2.1.1), ovšem v každé dimenzi.

Vícedimenzionální konvoluce je obdobná s jednodimenzionální :

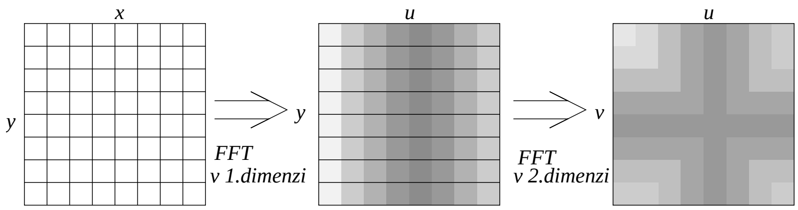
$$y(t_1,\dots,t_M)=h(\vec{t})*g(\vec{t})=\iint_{-\infty}^{\infty}h(t_1,\dots,t_M)g(t_1-\tau_1,\dots,t_M-\tau_M)\mathrm{d}\tau_1\dots\mathrm{d}\tau_M \tag{2.18}$$

Vícedimenzionální DFT (a inverzní DFT) je separabilní, tedy lze přepsat M dimenzionální DFT

$$H_{n_1,\dots,n_M}=\sum_{k_M=0}^{N_M}e^{\frac{-i2\pi n_Mk_M}{N_M}}\left(\sum_{k_{M-1}=0}^{N_{M-1}}e^{\frac{-i2\pi n_{M-1}k_{M-1}}{N_{M-1}}}\left(\dots\left(\sum_{k_1=0}^{N_1}h_{k_1,\dots,k_M}e^{\frac{-i2\pi n_1k_1}{N_1}}\right)\dots\right)\right) \tag{2.19}$$

Inverzní DFT lze rozepsat obdobným způsobem.

Ze separability plyne, že lze provést vícedimenzionální DFT, postupným výpočtem jednodimenzionálních DFT v každé dimenzi. Tedy k výpočtu vícedimenzionálního FFT můžeme využívat jednodimenzionálního FFT a transpozicí (pro větší efektivitu se ovšem provádí jednodimenzionální FFT na všech odpovídajících datech současně a transpozice se nahrazuje prostým přeindexováním).



Obrázek 2.8: Postup výpočtu FFT dvoudimenzionálních dat.

Vlastnosti vícedimenzionálních DFT jsou stejné jako v kap. 2.1.2, s tím rozdílem že veškeré skalární indexy přejdou ve vektory.

Zajímavá z hlediska výpočtu reálné FFT je skutečnost že symetrie reálných dat zůstává, opět s tím rozdílem, že skalární indexy přejdou ve vektory. Tedy spektrum reálných vícedimensionálních dat je periodické v každé ze svých dimenzí. Této vlastnosti lze opět využít, tedy v první dimenzi provedeme obdobné přeuspořádání dat (sudá data do reálné části, lichá data do imaginární části).

Data se tedy zkrátí v první dimenzi na polovinu. Na takto přeuspořádaná data aplikujme komplexní vícedimenzionální FFT.Poté jsou aplikovány rovnice (2.12), (2.13).

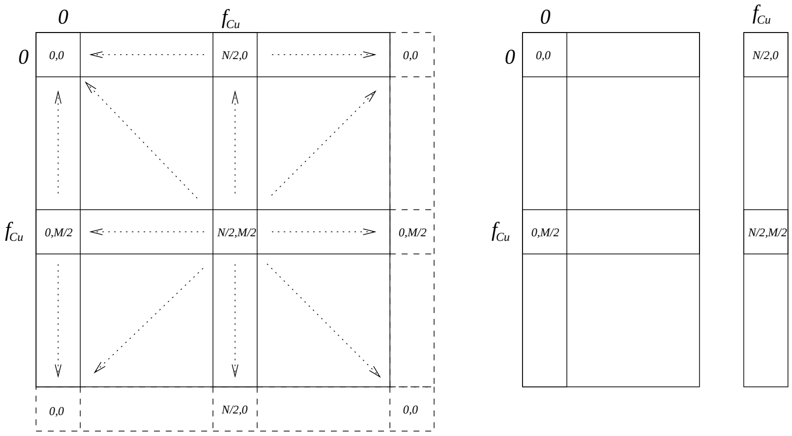
Data jsou v paměti uložena obdobně jako u jednodimenzionálního případu : jen polovina dat

$$H_{n_1,n_2,\ldots,n_M}, \quad n_1 \in \left[0, \frac{N_1}{2}\right] \quad n_2 \ldots n_M \in [0, N_{2,\ldots,M}]$$

a dále je nutné uložit celou Nyquistovu frekvenci v první dimenzi

$$H_{\frac{N_1}{2}+1,n_2,\ldots,n_M}, \quad n_2 \ldots n_M \in [0, N_{2,\ldots,M}]$$

Inverzní vícedimenzionální reálné FFT lze popsat obdobným způsobem.



Obrázek 2.9: Uložení DFT reálných dvoudimenzionálních dat.

2.2 Diskrétní sinová a cosinová transformace

Diskrétní Fourierova transformace předpokládá, že data se opakují ve všech dimenzích. Pokud rošíříme data určitým způsobem tak, aby vznikla data symetrická, pak Fourieorva transformace přejde v transformaci jinou. Symetrie můžeme popsat podle okrajových podmínek [4]. Typickými okrajovými podmínkami jsou :

- mají nulové hodnoty na okrajích $f(0) = 0$ (Dirichletova podmínka)
- mají nulové derivace na okrajích $f'(0) = 0$ (Neumannova podmínka)

V první případě je přirozenou transformací transformace *sinová*, v druhém pak transformace *cosinová.*

Tyto transformace mají každá jedinou spojitou definici, ovšem v diskrétní podobě těchto transformací je možné chápat výše uvedené okrajové podmínky různými způsoby (navíc je nutné v diskrétním případě specifikovat i okrajové podmínky na opačném konci intervalu). Je tedy možné dle typů okrajových podmínek pracovat s různými podobami těchto diskrétních transformací.

Okrajové podmínky na konci intervalu mohou být dvojí: Neumannova či Dirichletova. Každá okrajová podmínka má v diskrétním případě dvě aproximace : centrování na celý vzorek či na půl vzorku. Tedy sinová a cosinová transformace mají každá 8 diskrétních podob. Důležité jsou

[4] Kromě zpracování signálů a obrazových dat lze použít Fourierovu transformaci k řešení diferenciálních (diferenčních) rovnic.

především transformace, u kterých se shoduje způsob centrování na konci a na začátku intervalu. Všechny tyto transformace jsou lineární a ortogonální.

Nejdůležitější z hlediska zpracování obrazu je diskrétní cosinová transformace (DCT) respektive DCT2, kde je centrování na půl vzorku na obou krajích a na pravém konci signálu je Neumannova okrajová podmínka $u'(\pi) = 0$.

2.2.1 Diskrétní cosinová transformace DCT

DCT (DCT2) je definována,

$$H_n = C_n \sum_{k=1}^{N-1} h_k \cos\left(\frac{\pi k(n+\frac{1}{2})}{N}\right) \tag{2.20}$$

, kde

$$C_n = \begin{cases} \frac{1}{\sqrt{N}} & n = 0 \\ \sqrt{\frac{2}{N}} & n \neq 0 \end{cases} \tag{2.21}$$

Inverzní DCT je definována,

$$h_k = \sum_{n=1}^{N-1} C_n H_n \cos\left(\frac{\pi k(n+\frac{1}{2})}{N}\right) \tag{2.22}$$

Tato transformace má jako bázi kompletní sadu kosínů přes celou jednu polovinu periody. Tato DCT2 je nejpopulárnější ze všech, protože obsahuje DC složku (pro $n = 0$ je bázový vektor $(1, 1, \dots, 1)$) a velmi mnoho informace je obsaženo na samém začátku intervalu.

Způsoby výpočtu diskrétní cosinové transformace

Diskrétní cosinová transformace může být vypočítána skrze Fourierovu transformaci a to dvěmi metodami.

První metoda čerpá z Fourierovi transformace dat s určitou symetrií, druhá metoda spočívá v použití pomocných dat, která vytvoříme ze vstupních dat a jejichž Fourierova transformace má vlastnosti, které lze využít k získání této diskrétní cosinové transformace.

Často se používá DCT jen na malé části dat (obvykle velikosti 8). Data jsou rozděleny do bloků po 8 hodnotách[5], a poté je na tyto bloky provedena vysoce optimalizovaná DCT. Tato metoda je především užitečná pro kompresi obrazu, využívají jí JPEG a MPEG standarty. DCT je v tomto případě odvozována buď přímo z definice DCT, či je odvozena z výpočtu skrze DFT.

Díky ortogonalitě stačí k výpočtu inverzní DCT obrátit algoritmus pro výpočet DCT (conjugovaná transpozice transformační matice DCT).

V dalším textu se budeme zabývat těmito metodami.

Metoda využívající symetrie k výpočtu cosinové transformace

Rozšíříme data na dvojnásobek jejich délky tak, aby jsme získali *sudou* funkci okolo $k = N - \frac{1}{2}$ a $k = -\frac{1}{2}$ (tedy centrvái na půl vzorku na obou koncích),

$$h_{2N-k-1} = h_k \qquad k = 0, \dots, N-1 \tag{2.23}$$

Fourierovu transformaci takto rozšířených dat můžeme vyjádřit,

$$H_n = \sum_{k=0}^{2N-1} h_k e^{\frac{-i2\pi kn}{2N}} = \dots = 2e^{i\frac{\pi n}{2N}} \sum_{k=0}^{N-1} h_k \cos\left(\frac{\pi n\left(k+\frac{1}{2}\right)}{N}\right)$$

Až na činitel $2e^{i\frac{\pi n}{2N}}$ jsme získali z Fourierovi transformace takto rozšířených dat diskrétní cosinovou transformaci dat původních.

[5] ve vícedimenzionálním případě pak do bloků, jež mají v každé dimenzi 8 hodnot

0							N−1
0	1	2	3	4	5	6	7

0								N							2N−1
0	1	2	3	4	5	6	7	7	6	5	4	3	2	1	0

Obrázek 2.10: Symetrie použitá k výpočtu DCT

Metoda výpočtu skrze pomocnou funkci

Pro vytvoření pomocných dat využijeme funkci,

$$y_k = \frac{1}{2}(h_k + h_{N-k-1}) - \sin\left(\frac{\pi\left(k+\frac{1}{2}\right)}{N}\right)(h_k - h_{N-k-1}) \qquad k = 1, \ldots, N-1 \tag{2.24}$$

Pole takto vytvořené má stejné rozměry jako původní data. Pokud na tato data aplikujeme Fourierovu transformaci, pak lze výslednou reálnou a imaginární část Fourierova obrazu vyjádřit,

$$R_n = \cos\left(\frac{\pi k}{n}\right) H_{2n} + \sin\left(\frac{\pi k}{n}\right)(H_{2n+1} - H_{2n-1}) \tag{2.25}$$

$$I_n = \sin\left(\frac{\pi k}{n}\right) H_{2n} - \cos\left(\frac{\pi k}{n}\right)(H_{2n+1} - H_{2n-1}) \tag{2.26}$$

Proto H_n lze vypočítat z Fourierovi transformace pomocných dat diskrétní cosinovou transformaci,

$$H_{2n} = \cos\left(\frac{\pi k}{n}\right) R_n + \sin\left(\frac{\pi k}{n}\right) I_n \tag{2.27}$$

$$H_{2n-1} = \sin\left(\frac{\pi k}{n}\right) R_n - \cos\left(\frac{\pi k}{n}\right) I_n + H_{2n+1} \tag{2.28}$$

Sudé členy lze vypočítat přímo, ovšem k výpočtu lichých členů je třeba použít rekurzi. Rekurzi zahájíme v $n = \frac{N}{2} - 1$ a použijeme $H_{N-1} = -H_{N+1}$. Pak z (2.28) plyne :

$$H_{N-1} = \frac{1}{2} R_{\frac{N}{2}}$$

Pro inverzní transformaci stačí uvedené kroky obrátit k získání inverzní transformace

Metoda výpočtu DCT8 z definice transformace

Definice transformace pro pevnou délku $N = 8$,

$$H_n = C_n \sum_{k=1}^{7} h_k \cos\left(\frac{\pi k(n+\frac{1}{2})}{8}\right) \tag{2.29}$$

Rozborem této rovnice, lze dojít k velmi efektivnímu algoritmu. Využijeme těchto symetrií :

$$\begin{aligned}
\cos(x) &= \cos(-x) \\
\cos(\pi + x) &= -\cos(x) \\
\cos(2\pi + x) &= \cos(x) \\
\cos\left(\frac{\pi}{2} + x\right) &= -\cos\left(\frac{\pi}{2} - x\right)
\end{aligned} \tag{2.30}$$

Ve výpočtu dále použijeme následujících konstant :

$$
C_k = \cos\left(\frac{k\pi}{16}\right) \quad S_k = \sin\left(\frac{k\pi}{16}\right) \qquad
\begin{array}{ll}
C_1 = S_7 = 0.9808 & C_5 = S_3 = 0.5556 \\
C_2 = S_6 = 0.9239 & C_6 = S_2 = 0.3827 \\
C_3 = S_5 = 0.8315 & C_7 = S_1 = 0.1951 \\
C_4 = S_4 = 0.7071 &
\end{array}
$$

Za použití symetrií (2.30) zapíšeme každou hodnotu výstupu z DCT,

$$
\begin{aligned}
2H_0 &= C_4\left(h_{07} + h_{34} + h_{16} + h_{25}\right) \\
2H_4 &= C_4\left(\left(h_{07} + h_{34}\right) - \left(h_{16} + h_{25}\right)\right) \\
\\
2H_2 &= C_6\left(d_{12} - d_{56}\right) + C_2\left(h_{07} - h_{34}\right) \\
2H_6 &= -C_2\left(d_{12} - d_{56}\right) + C_6\left(h_{07} - h_{34}\right) \\
\\
2H_1 &= C_1 d_{07} + C_7 d_{34} + C_3 d_{16} + C_5 d_{25} \\
2H_7 &= C_7 d_{07} - C_1 d_{34} - C_5 d_{16} + C_3 d_{25} \\
\\
2H_3 &= C_3 d_{07} - C_5 d_{34} - C_7 d_{16} - C_1 d_{25} \\
2H_5 &= C_5 d_{07} + C_3 d_{34} - C_1 d_{16} + C_7 d_{25}
\end{aligned}
$$

,kde

$$
d_{ij} = h_i - h_j \qquad h_{ij} = h_i + h_j
$$

Výpočet H_2 a H_6 obsahuje rotaci vektoru[6], jehož složky jsou $(d_{12} - d_{56})$ a $(h_{07} - h_{34})$. Rotaci lze provést rychleji,

$$
\begin{aligned}
x &= cos(\alpha)x + sin(\alpha)y\,(+\cos(\alpha)y - \cos(\alpha)x) = cos(\alpha)\,(x+y) + y\,(sin(\alpha) + cos(\alpha)) \\
y &= -sin(\alpha)x + cos(\alpha)y\,(+\cos(\alpha)x - \cos(\alpha)x) = cos(\alpha)\,(x+y) - x\,(sin(\alpha) + cos(\alpha))
\end{aligned}
$$

což jsou pouze 3 násobení a 3 sčítání, namísto 4 násobení a 2 sčítání.

Ve výpočtu H_1, H_7, H_3 a H_5 je možné provést úpravu, která sníží počet rotací ve výpočtu. Vezmeme-li poslední dvojčlen u H_3, $S_1 d_{16} + C_1 d_{25}$ a budeme se jej snažit nahradit $Ah_{12} + Bd_{12} + Ch_{56} + Dd_{56}$ pak:

$$
A = C_4 C_3 \qquad B = -C_4 S_3 \qquad C = -C_4 C_3 \qquad D = -C_4 S_3
$$

Rovnice pro výpočet H_3 bude tedy po takovéto úpravě vypadat,

$$
2H_3 = \left(C_3 d_{07} - S_3 d_{34}\right) - C_4\left(C_3\left(h_{12} - h_{56}\right) - S_3\left(d_{12} + d_{56}\right)\right)
$$

Obdobným způsobem můžeme upravit i H_5, H_1 a H_7 :

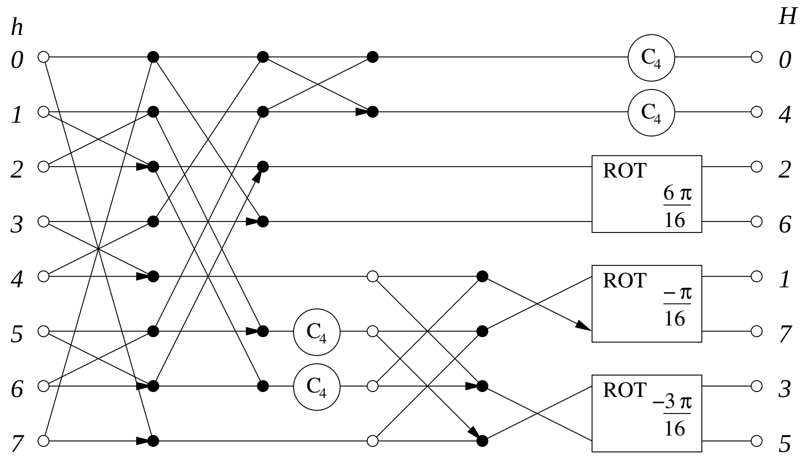
$$
\begin{aligned}
2H_5 &= \left(S_3 d_{07} + C_3 d_{34}\right) + C_4\left(-S_3\left(h_{12} - h_{56}\right) - C_3\left(d_{12} + d_{56}\right)\right) \\
2H_1 &= C_1\left(d_{07} + C_4\left(h_{12} - h_{56}\right)\right) - S_1\left(-d_{34} - C_4\left(d_{12} + d_{56}\right)\right) \\
2H_7 &= S_1\left(d_{07} + C_4\left(h_{12} - h_{56}\right)\right) + C_1\left(-d_{34} - C_4\left(d_{12} + d_{56}\right)\right)
\end{aligned}
$$

V případě H_3 a H_5 se tedy jedná o rotaci vektoru $[d_{07} - C_4\left(h_{12} - h_{56}\right)\ d_{34} - C_4\left(d_{12} + d_{56}\right)]$ o úhel $-\frac{3\pi}{16}$. U H_1 a H_7 jde o rotaci vektoru $[d_{07} + C_4\left(h_{12} - h_{56}\right)\ d_{34} + C_4\left(d_{12} + d_{56}\right)]$ o úhel $-\frac{\pi}{16}$.
Inverzní DCT lze provést díky ortogonalitě transformace pouhým otočením grafu (transpozicí transformační matice).

Metoda výpočtu DCT8 "skrze DFT"

Tento způsob výpočtu je součástí standartu JPEG a MPEG. Tento výpočet je nejefektivnější ze všech, jelikož velká část násobení je až na konci samotné transformace a je závislá jen na poloze

[6] C_2 odpovídá S_6



Obrázek 2.11: Výpočetní graf pro DCT velikosti 8 odvozen z definice DCT

ve výstupu, tudíž toto násobení může být součástí kvantizace.

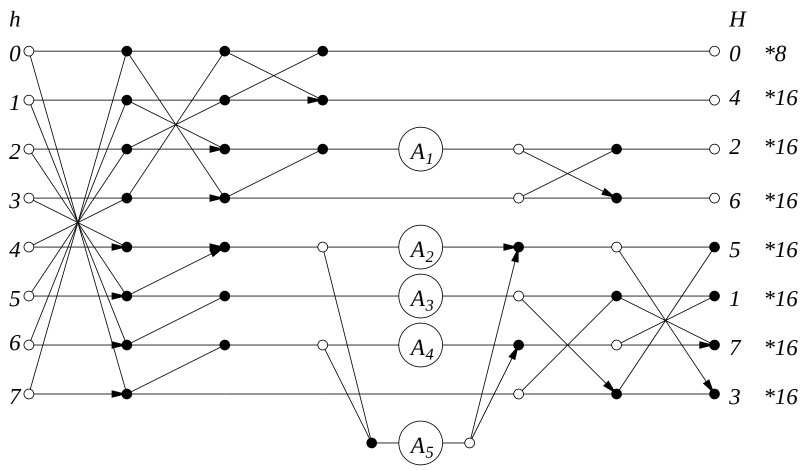
Prozatím můžeme "přehlédnout" koeficienty C_u v definici DCT2, jelikož tyto koeficienty budou též součástí kvantizace. Opět provedeme rozšíření dat, ovšem tentokrát pouze ve smyslu transformace. Fouriova transformace rozšířených dat viz. (2.23) dá,

$$\mathcal{RE}\left\{F_n\right\} = 2\cos\left(\frac{\pi n}{16}\right)\sum_{k=0}^{7} h_k \cos\left(\frac{\pi n(2k+1)}{16}\right) \tag{2.31}$$

Člen $2\cos\left(\frac{\pi n}{16}\right)$ se opět stane součástí kvantizace. Odvodíme tedy výpočet DCT z výpočtu DFT dvojnásobné délky. Rozložením transformační matice pro reálnou část DFT dostaneme.

$$\begin{aligned}
m_1 &= 2\left[(h_{07}+h_{34})+(h_{07}+h_{16})\right] & s_5 &= m_4+m_6\\
m_2 &= (h_{07}+h_{34})-(h_{25}+h_{16}) & s_6 &= m_4-m_6\\
m_3 &= (h_{07}-h_{34}) & s_7 &= m_8-m_7\\
m_4 &= d_{07} & s_8 &= m_9-m_7\\
m_5 &= \cos\left(\tfrac{\pi}{4}\right)\left[(h_{16}-h_{25})+(h_{07}-h_{34})\right] & &\\
m_6 &= \cos\left(\tfrac{\pi}{4}\right)(d_{25}+d_{16}) & 16H_0 &= m_1\\
m_7 &= \cos\left(\tfrac{3\pi}{8}\right)\left[(d_{16}-d_{34})-(d_{25}+d_{34})\right] & 16H_4 &= m_2\\
m_8 &= \left[\cos\left(\tfrac{\pi}{8}\right)+\cos\left(\tfrac{3\pi}{8}\right)\right](d_{16}+d_{07}) & 16H_2 &= m_3+m_5\\
m_9 &= \left[\cos\left(\tfrac{\pi}{8}\right)-\cos\left(\tfrac{3\pi}{8}\right)\right](d_{25}+d_{34}) & 16H_6 &= m_3-m_5\\
& & 16H_3 &= s_6-s_8\\
& & 16H_5 &= s_6+s_8\\
& & 16H_1 &= s_5+s_7\\
& & 16H_7 &= s_5-s_7
\end{aligned}$$

Inverzní DCT provedeme pouhým obrácením výpočetního grafu.



Obrázek 2.12: Výpočetní graf pro DCT velikosti 8 odvozen z reálné části DFT symetrických dat

$$
\begin{aligned}
m_1 &= H_0 \\
m_2 &= H_4 \\
d_{0734} &= H_2 + H_6 \\
m_5 &= H_2 - H_6 \\
s_5 &= H_1 + H_7 \\
s_6 &= H_5 + H_3 \\
s_7 &= H_1 - H_7 \\
s_8 &= H_5 - H_3 \\
\\
d_{07} &= (s_5 + s_6) \\
m_6 &= (s_5 - s_6) \\
T &= 2\cos\left(\tfrac{3\pi}{8}\right)(s_8 - s_7) \\
d_{2345} &= 2\left[\cos\left(\tfrac{\pi}{8}\right) + \cos\left(\tfrac{3\pi}{8}\right)\right] s_8 - T \\
d_{0167} &= 2\left[\cos\left(\tfrac{\pi}{8}\right) - \cos\left(\tfrac{3\pi}{8}\right)\right] s_7 - T
\end{aligned}
$$

$$
\begin{aligned}
s_{0347} &= m_1 + m_2 \\
s_{1256} &= m_1 - m_2 \\
d_{16} &= d_{0167} - d_{07} \\
d_{1256} &= m_6 / \cos\left(\tfrac{\pi}{4}\right) \\
d_{25} &= d_{1256} - d_{16} \\
d_{34} &= d_{2345} - d_{25} \\
s_{07} &= d_{0734} + s_{0347} \\
s_{34} &= s_{0347} - d_{0734} \\
d_{1625} &= m_5 / \cos\left(\tfrac{\pi}{4}\right) - d_{0734} \\
s_{16} &= d_{1625} + s_{1256} \\
s_{25} &= s_{1256} - d_{1625}
\end{aligned}
$$

2.2.2 Konvoluce v DCT oblasti

Konečné vektory h a x jsou symetricky prodlouženy na délku $2N$ či $2N-1$, kde N je délka vektorů. Tyto jsou pak konvolvovány obvyklou cyklickou cestou tedy skrze DFT 2 násobné délky. Poté je výstup omezen na originální počet N vzorků.

Tato *symetrická* konvoluce,

$$h *_s x$$

odpovídá v DCT oblasti násobení cosinových dat.

2.2.3 Vícedimenzionální DCT

Ve vícedimensionálním případě má DCT separabilní jádro,

$$H_{n_1,n_2} = C_{n_1} C_{n_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} h_{k_1,k_2} \cos\left(\frac{\pi n_1(2k_1+1)}{2N_1}\right) \cos\left(\frac{\pi n_2(2k_2+1)}{2N_2}\right)$$

Lze tudíž provést transformaci vícedimenzionálních dat, postupnou aplikací jednodimensionální transformace v každé dimenzi. Princip symetrické konvoluce platí také ve vícedimensionálním případě.

2.3 Hartleyova transformace

Hartleyovu transformaci předložil R.N.Bracewell jako náhradu DFT z hlediska konvoluce. Diskrétní Hartleyova transformace (DHT) vychází z DFT a přináší řadu výhod.

Přestože je DHT definováno pro reálná i komplexní data, její praktická hodnota vychází z výhody symetrie DFT reálných dat. Proto budeme dále uvažovat jen reálná data. Reálná data o N vzorcích mají DFT, která obsahuje N komplexních hodnot ($2N$ reálných hodnot). Ovšem pouze N z těchto hodnot je nezávislých (více viz. kap. 2.1.3). Hartleyova transformace je jednou z možností jak využít této redundance. Protože reálná složka DFT reálných dat je sudá a imaginární složka DFT reálných dat je lichá, můžeme je spojit dohromady s vědomím, že mohou být opět rozdělena na plně komplexní DFT, kdy bude potřeba.

Definujeme proto jednodimenzionální diskrétní Hartleyovu transformaci reálných dat jako reálnou část DFT mínus imaginární část DFT:

$$H_n = \mathcal{RE}\left\{F_n\right\} - \mathcal{IM}\left\{F_n\right\} \tag{2.32}$$

Jestliže tento vztah dosadíme do rovnice DFT (2.5) dostaneme Hartleyovu transformaci:

$$H_n = \sum_{k=0}^{N-1} h_k \left[\cos\left(\frac{2\pi kn}{N}\right) + \sin\left(\frac{2\pi kn}{N}\right)\right] = \sum_{k=0}^{N-1} h_k \mathrm{cas}\left(\frac{2\pi kn}{N}\right) \tag{2.33}$$

,kde

$$\mathrm{cas}\left(\frac{2\pi kn}{N}\right) = \cos\left(\frac{2\pi kn}{N}\right) + \sin\left(\frac{2\pi kn}{N}\right) \tag{2.34}$$

Inverzní transformace pak je:

$$h_k = \frac{1}{N}\sum_{n=0}^{N-1} H_n \mathrm{cas}\left(\frac{2\pi kn}{N}\right) \tag{2.35}$$

2.3.1 Rychlá Hartleyova transformace FHT

Bracewell a ostatní ukázali, že rychlá DHT (FHT) překoná FFT co do rychlosti a navíc má výhodu menšího velikosti transformovaných dat, díky tomu že transformuje reálná data opět na reálná.

Stejně jako u FFT vychází FHT z principu postupného rozdělování intervalu, z čehož plyne, že opět lze takto transformovat data pouze délky mocniny dvou. Základem FHT je následující vztah:

$$H_n = H_n^e + \cos\left(\frac{2\pi f}{N}\right) H_n^o + \sin\left(\frac{2\pi f}{N}\right) H_{N-n}^o \tag{2.36}$$

kde H_n^e je HT vektoru všech hodnot se sudými indexy a H_n^o je HT vektoru všech hodnot s lichými indexy. Konec této rekurence je doplněn faktem, že HT vektoru jen s jednou hodnotou je tento vektor[7]. Stejně tak jako u FFT je obvyklým způsobem výpočtu DHT rozdělení výpočtu na dvě části: přeskupení hodnot ve vektoru a vlastní výpočet "buttefly".

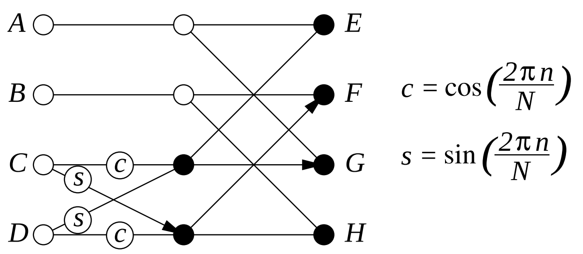
Způsob přeskupení hodnot je stejný jako u FFT : bitový reverz indexů hodnot. Po přeskupení hodnot obsahuje vektor N Hartleyových transformací vektorů délky 1. Následující část bere vždy dvě "menší" transformace a skombinuje je do transformace dvojnásobné délky. V $log(n)$ krocích je proto N malých transformací délky 1 kombinváno do kompletní transformace původního vektoru.

Je nutné upozornit na některá fakta týkající se výpočtu butterfly: Díky periodicitě transformace platí následující vztahy pro každé $n \in [0, N-1]$:

$$H^e_{\frac{N}{2}+n} = H^e_n \qquad H^o_{\frac{N}{2}+n} = H^o_n$$

Buttefly je nutno počítat ze čtyřech hodnot H_n, $H_{\frac{N}{2}-n}$, $H_{\frac{N}{2}-n}$, H_{N-n}, jelikož se snažíme provádět transformaci s výpočtem v místě. verze

[7]Je vidět že situace je více nežli podobná situaci u FFT



Obrázek 2.13: Výpočetní jednotka rychlé Hartleyovi transformace FHT

2.3.2 Konvoluce v Hartleyově oblasti

Konvoluční formule v Hartleyově oblasti je:

$$H_n = \frac{1}{2}\left[X_n\left(Y_n + Y_{-n}\right) + X_{-n}\left(Y_n - Y_{-n}\right)\right] \tag{2.37}$$

pokud je ovšem Y_n sudé pak konvoluce bude:

$$H_n = X_n Y_n \tag{2.38}$$

Pokud je tedy Y_n sudé, pak je konvoluce rychlejší v Hartleyově oblasti nežli ve Fourierově oblasti.

2.3.3 Vícedimensionální diskrétní Hartleyova transformace

Vícedimensionální DHT je definováno podobně jako DFT:

$$H_{n_1,\ldots,n_M} = \sum_{k_1=0}^{N_1-1} \cdots \sum_{k_2=0}^{N_M-1} h_{k_1,\ldots,k_M} \operatorname{cas}\left(2\pi\left(\frac{k_1 n_1}{N_1} + \ldots + \frac{k_M n_M}{N_M}\right)\right) \tag{2.39}$$

Bohužel vícedimenzionální DHT není přímo separabilní, tak jako FT. Bracewell ovšem popsal korekční mechanismus, k překlenutí chybějící separability HT. Je založen na vztahu:

$$\begin{aligned} 2\operatorname{cas}(\alpha+\beta) &= \operatorname{cas}(\alpha)\operatorname{cas}(\beta) + \\ &+ \operatorname{cas}(\alpha)\operatorname{cas}(-\beta) + \\ &+ \operatorname{cas}(-\alpha)\operatorname{cas}(\beta) - \\ &- \operatorname{cas}(-\alpha)\operatorname{cas}(-\beta) \end{aligned} \tag{2.40}$$

S využitím tohoto vztahu jsou nejdříve data transformována FHT jako kdyby se jednalo o separabilní bázi pomocnou transformací:

$$H_{n_1,\ldots,n_M} = \sum_{k_1=0}^{N_1-1} \cdots \sum_{k_2=0}^{N_M-1} h_{k_1,\ldots,k_M} \operatorname{cas}\left(\frac{2\pi k_1 n_1}{N_1}\right) \ldots \operatorname{cas}\left(\frac{2\pi k_M n_M}{N_M}\right) \tag{2.41}$$

Poté je provedena korekce dat pomocí (2.40). Tato korekce může být provedena inplace s použitím buttefly, které provede čtyři kombinace v jednom kroku.

Hong, Hang a Rao optimalizovali korekční mechanismus pro třídimenzionální případ[8]. Dospěli ke vztahu, který kombinuje opět pouze čtyři hodnoty místo předpokládaných osmi hodnot:

$$\begin{aligned} 2\operatorname{cas}(\alpha+\beta+\gamma) &= \operatorname{cas}(-\alpha)\operatorname{cas}(\beta)\operatorname{cas}(\gamma) + \\ &+ \operatorname{cas}(\alpha)\operatorname{cas}(-\beta)\operatorname{cas}(\gamma) + \\ &+ \operatorname{cas}(-\alpha)\operatorname{cas}(\beta)\operatorname{cas}(-\gamma) - \\ &- \operatorname{cas}(-\alpha)\operatorname{cas}(-\beta)\operatorname{cas}(-\gamma) \end{aligned} \tag{2.42}$$

[8] Jinak by bylo nutné aplikovat korekční mechanismus postupně v průběhu transformace

Také tento vztah může být využit ke korekci pomocné transformace na Hartleyovu transformaci.

2.4 Využití diskrétních transformací ke zpracování digitalizovaného obrazu

2.4.1 Konvoluce

Konvoluce je hlavní silou diskrétních transformací. Díky velmi rychlým implementacím je ve většině případech[9] konvoluce skrze frekvenční oblast mnohokrát rychlejší, nežli konvoluce v datové oblasti.

Filtrace

Konvoluce se především využívá k filtraci. Filtrací rozumíme operaci, při které úmyslně zvýrazňujeme či zanedbáváme některé frekvence. Nejběžnější formou filtrací jsou takzvané Low-Pass (LP), High-Pass (HP) a Band-Pass (BP) filtry.

Low-Pass filtry slouží k rozmazání (vyhlazení) signálu, obrázku či trojrozměrného objektu.

High-Pass filtry slouží k detekci hran v obrazu.

Band-Pass filtry se používají především ve zpracování signálu, kde potřebuje jen určitou část spektra.

Obecně se za pomoci filtrů snažíme v datech zvýraznit některé jejich vlastnosti, či jiné potlačit.

K filtraci lze použít, různých druhů filtrů: obdélníkový, Butterworth apod.

Obdélníkový filtr ořízne všechny frekvence nalézající se mimo daný interval.

Butterworth filtr oproti obdélníkovému filtru neprovede hrubý ořez frekvencí, ale raději provede plynulé postupné oříznutí frekvencí mimo daný interval. Ovšem oproti obdélníkovému filtru má tento typ filtru delší dobu provádění.

Restaurace obrazu

Velmi rozsáhlé využití konvoluce (a diskrétních transformací) nalezneme u *restaurace obrazu*. Jedná se techniku předzpracování obrazu, která se snaží potlačit porušení obrazu na základě znalosti poruchy nebo jejího odhadu. Většina metod pro restauraci obrazu využívá Fourierova spektra celého obrazu.

Příčiny porušení obrazu mohou být různé:

- vada optické soustavy
- nelinearita optickoelektrického čidla
- nelinearita nebo zrnitost filmového materiálu
- vzájemný pohyb snímače obrazu a předmětu
- nevhodné zaostření snímače
- turbulence atmosféry

Postupy restaurace obrazu mohou být *deterministické* či *stochastické*.

[9] pokud konvolujeme velmi rozsáhlá data společně s velmi malými daty, vyplatí se provést konvoluci v datové oblasti

Deterministický přístup se hodí pro obrazy bez šumu. Tyto metody vypočítají původní obraz pomocí transformace inverzní k degradační transformaci. Jednou z deterministických metod je *inverzní filtrace.*

Stochastický přístup použijeme, pokud je obraz zašuměn. Pro obnovení se hledá nejlepší filtr pomocí *metod stochastických modelů.* Kritériem optimality těchto modelů je některá míra odlišnosti původního obrazu, který byl odhadnut metodou obnovení. Jednou z velmi používaných metod je Wienerův filtr.

K popisu degradace obrazu může použít tento model :

$$g(x,y) = \iint_{a,b\in\mathcal{O}} f(a,b)\,h(x-a,y-b)\,\mathrm{d}a\mathrm{d}b + \nu(x,y)$$

,kde $g(x,y)$ je degradovaný obraz, $f(x,y)$ je původní obraz a $\nu(x,y)$ je aditivní šum.

Po Fourierově transformaci dostaneme :

$$G(u,v) = F(u,v)\,H(u,v) + N(u,v)$$

Je tedy zřejmé, že původní obraz $f(x,y)$, respektive jeho Fourierovu transformaci mohu získat dekonvolucí. Tato metoda se nazývá *inverzní filtrace* a lze ji zapsat

$$F(u,v) = G(u,v)\,H^{-1}(u,v)) - N(u,v)\,H^{-1}(u,v)$$

Hlavní část metody spočívá v určení degradační transformace[10]. Například pro degradaci způsobenou vzájemným pohybem kamery a objektu použijeme degradační funkci

$$H(u,v) = \frac{\sin(\pi V T u)}{\pi V u}$$

,kde V je rychlost vzájemného pohybu kamary a objektu a T je doba expozice. Tento vztah platí za předpokladu, že vzájemný pohyb je rovnoměrný a pouze ve směru x. Dalším příkladem může být degradace obrazu způsobená špatným zaostřením kamery. Va tomto případě je degradační transformace

$$H(u,v) = \frac{J_1(u,v)}{ar}$$

, kde a je vzdálenost správného zaostření a J_1 je Besselova funkce i prvního řádu $r^2 = u^2 + v^2$. Nevýhodou metody inverzní filtrace je nutná znalost šumu. Často se ovšem zašumění zanedbává.

Při stochastickém přístupu využijeme Wienerova filtru. Předpokládáme, že neznáme šum, ale známe jen jeho statistické parametry, tedy provádíme odhad $\hat{f}(x,y)$, snažíme se najít optimální filtr. Na základě metody nejmenších čtverců tedy dostaneme

$$H_w(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + \frac{S_{\nu\nu}(u,v)}{S_{gg}(u,v)}}$$

2.4.2 Korelace

Korelace je hledání statisticky významných závislostí mezi událostmi. Je definována

$$\mathcal{CORR}\{h(t),g(t)\} = \int_{-\infty}^{\infty} h(\tau)g(t+\tau)d\tau \tag{2.43}$$

Obě funkce $h(t)$, $g(t)$ jsou reálné. Pokud obsahují související událost budou mít v korelační funkci $\mathcal{CORR}\{h(t),g(t)\}$ špičku. Umístění této špičky odpovídá zpoždění posunutí mezi signály.

[10] obvykle jejího Fourierova obrazu

Pokud funkce obsahují společnou frekvenční složku objeví se vrchol ve Fourierově transformaci korelační funkce:

$$\mathcal{FT}\{\mathcal{CORR}\{h(t), g(t)\}\} = H(f)G^*(f)\mathcal{CORR}\{h(t), g(t)\} = h(t) * g(-t) \tag{2.44}$$

Pokud počítáme korelaci totožných dat pak se jedná o **autokorelaci** $\mathcal{CORR}\{h(t), h(t)\}$ signálu $h(t)$ a rovná se Fourierově transformaci výkonnostní spektrální intenzity. Autokorelace se sepoužívá na analýzu šumu.

V jednodimenzionálním případě (zvuk, vlnění) se korelace používá k porovnání signálů, repsektive ke zjištění souvislosti mezi signály. Při zpracování digitalizovaného obrazu nám může korelace pomoci při porovnávání obrazů.

$$\mathcal{CORR}\{h(x, y), g(x, y)\} = \iint_{-\infty}^{\infty} h(\tau_1, \tau_2) g(x + \tau_1, y + \tau_2) \mathrm{d}\tau_1 \mathrm{d}\tau_2 \tag{2.45}$$

2.4.3 Komprese

Komprese patří k nižší úrovni zpracování obrazu. Z hlediska diskrétních transformací je důležitá především *ztrátová komprese*, tedy komprese, kde se některé části obrazu zanedbávají. Ztrátová komprese se provádí v některé frekvenční oblasti a je založená na faktu, že lidské oko je citlivější na nižší frekvence v obrazu. Vysoké frekvence jsou tedy více či méně potlačeny[11] a frekvenční obraz je pak komprimován pomocí některého z komprimačních algoritmů.
Tento způsob komprese je velmi účinný a používá se především k ukládání a přenosu fotografického materiálu a video sekvencí. Jedním z jejích typických představitelů je standart JPEG využívající DCT po blocích 8×8 na převod obrazových dat do frekvenční oblasti, kde se uskutečňuje kavntizace následovaná kompresí. Na podobném principu je založen i standart MPEG, ten je ovšem zaměřen na video sekvence.

[11] například pomocí vektorové kvantizace

Kapitola 3

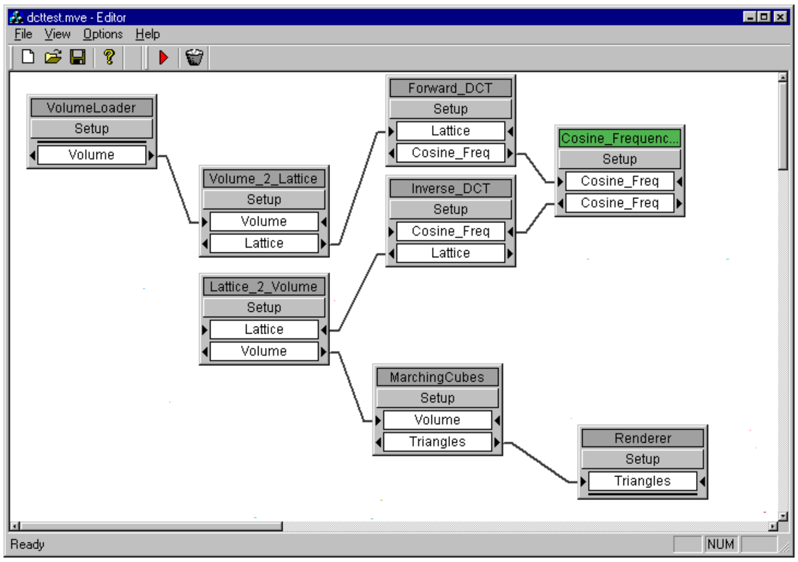
Praktická část

3.1 Způsob implementace diskrétních transformací

Uvedené transformace byly implementovány do modulárního visualizačního prostředí (MVE). MVE sestává z editoru a modulů.

Editor je grafické rozhraní, kde uživatel propojuje a nastavuje jednotlivé moduly, a které provádí veškerou obsluhu modulů.

Moduly jsou implementovány jako knihovny[1] a mohou být výpočetní, vstupní či výstupní. Speciálním oddílem výstupních modulů jsou renderery, které provádějí okamžité zobrazení dat na obrazovku, či jinou jejich interpretaci. Každý modul může mít nastavení a stav. Nastavení může uživatel měnit v editoru z dialogu poskytnutého knihovnou. Stav je uchováván editorem a ten jej předává modulu, který jej může libovolně měnit. Nastavení a stav nejsou u modulů povinné.



Obrázek 3.1: Editor MVE

[1]`dll` soubory pod Microsoft Windows

Editor komunikuje a obsluhuje knihovny pomocí volání funkcí modulů, a analýzou vrácených hodnot či dat. Veškerá data a parametry jsou předávány funkcím pomocí pevně daných datových struktur.

Moduly lze vytvářet v jakémkoliv vývojovém prostředí pod Microsoft Windows, nejlépe v jazycích C či Pascal. Moduly diskrétních transformací byly vytvářeny ve vývojovém prostředí Visual C++ 6.00.

V dalším textu jsou uvedeny pouze struktury a výpisy v jazyce C.

3.1.1 Datové struktury MVE, funkce knihovny

V datových strukturách se používají především typy definované v hlavičkovém souboru `Types.h` pro Pascal pak ve zdrojovém souboru `Types.pas`. Tyto soubory obsahují definice nových typů, kreré mají totožnou podobu v paměti jak v jazyce C tak i v jazyce Pascal. Tak se zajistí snadná přenositelnost dat mezi moduly napsanými v jazycích C a Pascal.

Editor používá pro komunikaci s modulem a předávání dat speciální struktury definované v hlavičkovém souboru `MVE_Include.h` případně `MVE_Include.pas`. Základní strukturou, která je modulu předána při zavolání výkonové funkce modulu je `DATA_DESCRIPTORS`. Tato struktura obsahuje tzv. *deskriptory*, malé struktury obsahující datové struktury popisující data[2], stav dat a typ dat.

`int`	`Num_Descr`	Počet obsažených descriptorů
`DATA_DESCR *`	`Descriptors`	Pole descriptorů dat
`int`	`Module_State`	Pro vracení stavu modulu (případné chyby)

Tabulka 3.1: Struktura předávaná modulu `DATA_DESCRIPTORS`

`int`	`Data_Type`	Typ obsažených dat
`void *`	`Data`	Ukazatel na data
`long`	`Data_Length`	Délka dat
`void *`	`Header`	Ukazatel na hlavičku s dalšímy informacemi
`long`	`Header_Length`	Délka hlavičky
`int`	`Data_State`	Aktuální stav dat

Tabulka 3.2: Deskriptor `DATA_DESCR`

Současně s touto strukturou je modulu ještě předáno nastavení a stav modulu.

Ke každému modulu musí být implementovány funkce, které umožňují prostřednictvím dialogového okna modul nastavit či uvolňují data modulem alokovaná. Dále musí být v knihovně implementovány funkce popisující jednotlivé moduly (jejich vstupy, výstupy atd.).

MVE totiž nejdříve připojí knihovnu a pomocí pevně dané funkce (`Get_Modules`) zjistí co za moduly daná knihovna poskytuje. Získá tak jméno modulu a podle typu operace (spusť modul, zruš výstupni data atd.) doplní za jméno modulu postfix daný druhem operace.

3.1.2 Obrazová a volumetrická data `T_Lattice_Data`

Základem k diskrétním transformacím jsou obrazová či volumetrická data. Informace o obrazových a volumetrických datech jsou uchovávány ve struktuře `T_Lattice_Data`.

[2] tyto struktury se liší dle typu dat

Tato struktura byla navržena tak, aby vyhovovala nejen diskrétním transformacím, ale též jiným modulům pracujícím s obrazovými a volumetrickými daty, například převod barevného modelu, prahování, segmentace apod.

Typ	Název	Stručný popis
`T_sInt`	`Res[3]`	Rozměry dat
`T_sFloat`	`Dis[3]`	Distorze dat
`void *`	`Data[LATTICE_MAXLAYERS]`	Datové prostory
`T_sInt`	`Cell_type[LATTICE_MAXLAYERS]`	Typy dat
`T_ColorModel`	`Color_model`	Barevný model
`char`	`Caption[255]`	Název dat
`char`	`Source[255]`	Zdroj dat
`T_sInt`	`Pos[3]`	Počáteční pozice dat
`T_sInt`	`Rot[3]`	Počáteční rotace dat
`T_Int`	`Min`	Minimální hodnota v datech
`T_Int`	`Max`	Maximální hodnota v datech
`T_Int`	`StartTreshold`	Začátek oříznutí

Tabulka 3.3: Datová struktura `T_Lattice_Data`

Tato struktura má 3 části :

Obecná část popisuje vlastnosti, které mají kterákoliv data obrazová či volumetrická. Do této části patří datové prvky `Res`, `Dis`, `Cell_type`.

Obrazová část obsahuje informaci o barevném modelu dat : datový prvek `Color_model`.

Volumetrická část obsahuje informace nutné k podrobnějšímu popisu a správnému zobrazení volumetrických dat. Jsou to datové prvky `Caption`, `Source`, `Pos`, `Rot`, `Min`, `Max`, `StartTreshold`.

Barevný model (`T_ColorModel`) je výčtový typ, obsahující nejpoužívanější barevné modely :

```
typedef enum { V,RGB,CMY,CMYK,HSV,YUV,YIQ } T_ColorModel;
```

Data jsou uložena neprokládaně, každá barevná složka v obrazu je uložena v jediném datovém prostoru[3]. Každá barevná složka může mít jiný typ dat(`Cell_type`). Jednotlivé typy jsou v tabulce 3.4.

Hodnota `Cell_type`	Datový typ
`CELLTYPE_byte`	`T_uByte`
`CELLTYPE_sbyte`	`T_Byte`
`CELLTYPE_word`	`T_sInt`
`CELLTYPE_long`	`T_Int`
`CELLTYPE_float`	`T_sFloat`
`CELLTYPE_double`	`T_Float`
`CELLTYPE_complex`	`T_Complex`

Tabulka 3.4: Datové typy v `Cell_type`

[3]Volumetrická data mají pouze jednu složku, což odpovídá barevnému modelu V

3.2 Implementace diskrétních transformací

K implementaci diskrétních transformací byl zvolen jazyk C. Celá implementace je rozdělena na tři části:

Implementace vlastních metod je programována ve standartu ANSI C, což zaručuje přenositelnost diskrétních transformací i na řadu jiných platforem nežli Microsoft Windows. Zdrojové kódy metod jsou umístěny v kořenovém adresáři zdrojových kódů[4].

Implementace rozhraní s MVE je programováno v Microsoft Visual C++ 6.00, a tato část je tudíž objektově orientována. Tato část je umístěna do podadresáře adresáře zdrojových kódů `win`. Obsahuje všechny soubory pro projekt v Microsoft Visual C++ 6.00.

Datové struktury MVE a makra jsou umístěny v samostatných podadresářích adresáře zdrojových kódů `mve` a `macros`. Mimoto je v adresáři `pascal` umístěn ekvivalent adresáře `mve` ovšem ve verzi pro jazyk `Pascal`.

Při programování je dodržována konvence týkající se parametrů funkcí a úklidu paměti. Parametry funkcí jsou implementovány v tomto pořadí : data (nejdříve vstup, pak výstup), poté rozměry (nejdříve rozměry poté mocniny dvou), datové typy, vše ostatní.

Pro úklid je používáno mechanismu implementovaného v souboru `macros/Error_Handling.h`, který pomocí maker při jakékoliv chybě umožňuje dealokovat vše co modul naalokoval a skončit "s čistým štítem". Všechna makra z tohoto souboru mají prefix `EH`.

Všechny moduly, které byly vytvořeny v rámci této diplomové práce jsou umístěny v knihovně `transform.dll`.

3.2.1 Rychlá Fourierova transformace a reálná rychlá Fourierova transformace

Datová struktura

Datová struktura pro frekvenční data[5], je podobná datové struktuře pro obrazová a volumetrická data.

Fourierova transformace a všechny operace ve frekvenci potřebují pro svojí činnost pouze rozměry dat, datové typy a data samotná. Ostatní informace obsažené ve struktuře jsou jen přenášeny pro další použití mimo frekvenci.

Je vidět, že se tato struktura liší od struktury pro obrazová a volumetrická data jen v několika datových prvcích.

`Original_Res` obsahuje rozměr obrazových dat, tedy dat před transformací. Protože jsou rozměry dat zvětšeny na nejbližší vyšší mocninu dvou, je nutné uchovávat původní velikost dat, protože na tuto velikost budou zpět transformovaná data oříznuta.

`Data` mohou být pouze komplexní, tudíž zde není na místě generický pointer.

`Fc` obsahuje uloženou Nyquistovu frekvenci v případě že bylo použito reálné FFT. Jinak jsou tato data prázdná, tedy pointer ukazuje na `NULL`.

`Original_cell_type` obsahuje datové typy původních dat. Typy původních dat budou totiž při zpětné transformaci opět obnoveny.

[4]tento adresář je umístěn na CD Referenční verze `CD:/private/transform`

[5]data tranformovaná Fourierovou transformací

Typ	Název	Stručný popis
T_sInt	Res[3]	Rozměry frekvenčních dat
T_sFloat	Dis[3]	Distorze půdovních dat
T_sInt	Original_Res[3]	Rozměry obrazových dat
P_Complex	Data[LATTICE_MAXLAYERS]	Datové prostory
P_Complex	Fc[LATTICE_MAXLAYERS]	Nyquistova frekvence
T_sInt	Original_cell_type[LATTICE_MAXLAYERS]	Typy dat
T_ColorModel	Color_model	Barevný model
char	Caption[255]	Název dat
char	Source[255]	Zdroj dat
T_sInt	Pos[3]	Počáteční pozice dat
T_sInt	Rot[3]	Počáteční rotace dat
T_Int	Min	Minimální hodnota v datech
T_Int	Max	Maximální hodnota v datech
T_Int	StartTreshold	Začátek oříznutí

Tabulka 3.5: Datová struktura frekvenčních dat `T_FreqFour_Data`

Výběr metod

Pro implementaci bylo vybráno FFT s decimací v čase z následujících důvodů:

1. Přeuspořádání lze provést již při kopírování dat do výstupního prostoru.
2. Jedná se o FFT s výpočtem v místě, tudíž není nutno alokovat zvláštní datový prostor. Bohužel v případě inverzního FFT je nutno tento prostor alokovat, jelikož frekvenční data na vstupu jsou pouze pro čtení. Nicméně veškeré inverzní transformace jsou implementovány tak, že datový prostor výstupu a výpočetní prostor jsou sdílené, pokud je velikost výpočetního prostoru větší nebo rovna velikosti výstupního prostoru. Tento případ nastává velmi často.

Výpočetní náročnost algoritmu

Již v teoretické části je rozebrána výpočetní náročnost FFT (kap. 2.1.3). Jelikož veškeré koeficienty $W = e^{\frac{-i2\pi}{N}}$ jsou rekurentně předpočítány je zdaleka nejnáročnější operací v algoritmu FFT násobení komplexních čísel ve výpočtu butterfly. Pro jednodimenzionální data velikost N je těchto násobení $\frac{N}{2}log_2N$. Ve vícedimenzionálním případě je aplikováno FFT na každý řádek, pak na každý sloupec atd. Takže například pro dvoudimenzionální FFT dat rozměrů N a M je vykonáno $\frac{NM}{2}\left(log_2N + log_2M\right)$ komplexních násobení. Připomeňme, že "obyčejné" DFT by vykonalo $NM\left(N+M\right)$ komplexních násobení.

V případě reálného FFT je množství komplexních násobení ještě menší. Jelikož jsou data přeskupena na poloviční délku je počet komplexních násobení ve výpočtovém grafu $\frac{N}{4}\left(log_2\left(N\right)-1\right)$. Je třeba však ještě připočíst poslední krok výpočtu buttefly FFT s decimací v čase tedy $\frac{N}{2}$.

Reálné FFT je tedy přibližně o polovinu rychlejší, nežli obyčejné FFT.

Implementace metod

Rychlá Fourierova transformace i reálná rychlá Fourierova transformace jsou implementovány ve zdrojových souborech `transform_fourier.c` a `transform_fourier.h`.

Hlavními částmi transformace jsou funkce `fft` a `ifft`, které zajišťují veškerou obsluhu výpočtu FFT. V těchto funkcích je nejprve zjištěna relevantnost dat, tedy jestli se jedná skutečně o data typu $DT\_LATTICE$[6], dále za pomoci funkce `makelogs` zjistí nejbližší vyšší mocniny dvou z

[6] čísla přiřazená jednotlivým typům dat jsou uvedena v hlavičkovém souboru `mve/Descrip_Types.h`

rozměrů dat.

Poté jsou podle nastavení alokovány alokovány paměťové prostory. V případě "obyčejného" FFT jsou alokovány prostory odpovídající celému spektru dat. Pokud je dovoleno a je možné provést reálné FFT jsou tyto prostory pouze poloviční velikosti (viz. kap. 2.1.3 a 2.1.4), ale je třeba také uložit Nyquistovu frekvenci celých dat pro první dimenzi.

K vnitřnímu popisu dat (adresa dat, rozměry dat, typ dat) uvnitř těchto funkcí se používá pomocných datových struktur `T_DDATA` a `T_DSIZE` definovaných v `utils_data.h`. Tyto datové struktury jsou navrženy všeobecně pro všechny diskrétní transformace i filtrace. Tyto struktury zpřehledňují program a výsledný kód se lépe odlaďuje.

Dále je vypočítána tabulka koeficientů FFT. Výpočet této tabulky provádí funkce `calc_tab`. Tabluka je počítána pomocí rekurentních vzorců rotace a je ukládána ve formě jednorozměrného pole. Toto pole obsahuje všechny hodnoty $W_N^n = e^{\frac{\pm i2\pi n}{N}}$ potřebné pro nadcházející FFT.

Dále funkce `reorder_and_copy` zajistí zkopírování dat do výstupního (či jen výpočetního) pro-

W_2^0	W_4^0	W_4^1	W_8^0	W_4^1	W_8^2	W_8^3	$\ldots$

Tabulka 3.6: Tabulka koeficientů FFT : $W_N^n = e^{-\frac{i2\pi n}{N}}$

storu zároveň s jejich přeuspořádáním (bitový reverz). Tato funkce je implementována pro dimenzi od jedné do tří. To jí umožňuje její rekurzivní implementace. V případě, že chceme provést pouze bitový reverz lze tuto funkci zavolat s nastaveným parametrem `inplace`.

Bitový reverz je někdy užitečné provádět pouze v některé z dimenzí. To se zejména využije při provádění FFT pro výpočet DCT. Směr bitového reverzu se nastavuje pomocí `direction`.

Bratrskou funkcí `reorder_and_copy` je funkce `reorder_and_copy_real`, která se používá při výpočtu reálného FFT. Tato funkce oproti `reorder_and_copy` ukládá data do výpočetního případně výstupního prostoru tak, jak je uvedeno v kap. 2.1.3, tedy že sudé vzorky vloží do reálné části dat a liché do imaginární části dat.

V tomto okamžiku jsou data připravena na výpočet butterfly, hlavní části výpočtu FFT. Výpočet buttefly provádí funkce `calc_buttefly_in_direction`, která provede výpočet všech butterfly v jednom směru. Pro výpočet butterfly ve všech směrech slouží funkce `calc_butterfly`.

V případě reálného FFT je potřeba ještě poslední krok FFT s decimací v čase dle rov. 2.12 a 2.13. Pokud se provádí reálné FFT přes všecny směry, použije se funkce `real_fft_last_step`. Tato funkce pracuje s daty, jaké jsou na obrázku 2.9. V případě že potřebujeme provést reálné FFT jen v jediném směru (tedy v případě výpočtu DCT), použije se funkce `fft_last_step_in_direction`. Tato funkce ukládá Nyquistovu frekvenci způsobem, který je na obrázku 2.7. Frekvenční data jsou uložená ve tvaru kterému se někdy říká *posunutý*, jelikož základní interval $-\frac{N}{2} \leq n \leq \frac{N}{2}$ je posunutý tak, aby počátek spektra[7] byl v počátku datového prostoru. Díky periodicitě spektra diskrétních dat se záporné frekvence, které se posunutím "schovali", objeví ve zbývajícím prostoru $\frac{N}{2} \leq n \leq N-1$.

3.2.2 Diskrétní cosinová transformace

Pro implementaci předpokládáme DCT jen z reálných dat.

Datová struktura

Tak jako u frekvenčních dat jsou cosinově transformovaná data uložena podobným způsobem: všechny operace nad cosinově transformovanými daty potřebují pro voji činnost pouze rozměry dat, data samotná a příznak zda se jedná o DCT po blocích velikosti 8 či o DCT přes celá data.

[7]rozuměj nulová frekvence

Zbývající část struktury je jen pro přenos informace pro další použití mimo DCT.

Typ	Název	Stručný popis
T_sInt	Res[3]	Rozměry DCT dat
T_sFloat	Dis[3]	Distorze půdovních dat
T_sInt	Original_Res[3]	Rozměry obrazových dat
P_sFloat	Data[LATTICE_MAXLAYERS]	Datové prostory
T_sInt	Original_cell_type[LATTICE_MAXLAYERS]	Typy dat
T_sInt	Dct8	Příznak DCT8
T_ColorModel	Color_model	Barevný model
char	Caption[255]	Název dat
char	Source[255]	Zdroj dat
T_sInt	Pos[3]	Počáteční pozice dat
T_sInt	Rot[3]	Počáteční rotace dat
T_Int	Min	Minimální hodnota v datech
T_Int	Max	Maximální hodnota v datech
T_Int	StartTreshold	Začátek oříznutí

Tabulka 3.7: Datová struktura pro data transformovaná DCT `T_FreqCos_Data`

Výběr metody

Pro výpočet DCT přes celá data byla vybrána metoda provádějící výpočet skrze pomocná data. Tato metoda má oproti metodě provádějící transformaci pomocí symetrie dvě výhody:

1. Lze jí provést v datovém prostoru se stejnou velikostí jako data. Naproti tomu transformace skrze symetrii potřebuje prostor dvakrát větší.

2. Je rychlejší nežli transformace skrze symetrii.

Pro výpočet DCT po blocích šířky 8 byla zvolena metoda odvozená od DFT. Tato metoda totiž umožňuje přesunout část násobení do kvantizačního bloku. I když neprovádíme žádnou kvantizaci, tato vlastnost je užitečná, neboť tato násobení sloučíme s násobením celého vektoru C_n, který by se musela data přesto vynásobit.

Výpočetní náročnosti metod

Obě metody výpočtu DCT celých dat počítají DCT skrze reálné FFT, jehož výpočetní náročnost je $\mathcal{O}\left\{\frac{N}{4}\left(log_2\left(N\right)-1\right)+\frac{N}{2}\right\}$. První metoda (metoda výpočtu skrze symetrii), potřebuje ke svému výpočtu dvojnásobně velký datový prostor a musí provést dodatečné násobení po FFT. Naproti tomu druhá metoda (metoda výpočtu skrze pomocnou funkci) potřebuje stejný prostor jako výstupní data, ovšem provádí dodatečné násobení jak před výpočtem FFT tak i po něm.

Výpočetní náročnost první metody je větší nežli u obyčejného FFT. U druhé metody jsou vytvářena pomocná data za pomocí dvou reálných násobení vždy pro dva vzorky dat. Komplexní násobení jsou čtyři reálná násobení a dva součty. Pokud součty zanedbáme, pak výpočetní náročnost vytváření pomocných dat je $N/4$. Výpočetní náročnost dovýpočtu DCT je $N/2$, jelikož se jedná vlastně také o butterfly. Tedy dohromady dostaneme výpočetní náročnost druhé metody $\mathcal{O}\left\{\frac{N}{2}+\frac{N}{4}log_2 N\right\}$. Druhá metoda je tedy rychlejší. .

Výpočet obou metod je zatížen faktem, že se musí výsledek násobit koeficientem C_n (viz. rov. 2.20), což zpomaluje výpočet DCT.

Výpočetní náročnost algoritmů pro výpočet DCT8 je pro oba algoritmy uvedené v teoretické části stejná : 13 násobení , 29 součtů. Metoda výpočtu odvozená od výpočtu DFT umožňuje, aby

8 násobení bylo součástí kvantizace.

Implementace vybrané metody

Implementace DCT přes celá data i DCT po blocích šířky 8 je umístěna ve zdrojových souborech `transform_cosine.c` a `transform_cosine.h`. Hlavními funkcemi jsou `dct` a `idct`, které opět řídí proces výpočtu DCT.

Stejně jako u FFT je zkontrolován typ dat na vstupu a alokován výstupní deskriptor a v případě DCT přes celá data jsou opět za pomoci funkce `makelogs` zjištěny nejbližší vyšší mocniny dvou z rozměr dat. V případě DCT po blocích šířky 8 jsou skrze funkci `make_sizes_mod_8` zjištěny nejbližší vyšší násobek 8.

Poté jsou alokovány výstupní datové prostory a pomocné datové prostory. Výstupní datové prostory jsou typu `T_sFloat`. U DCT přes celá data se musí alokovat pomocná data, kvůli rekurzi nutné k výpočtu DCT z reálného FFT pomocné funkce (2.28).

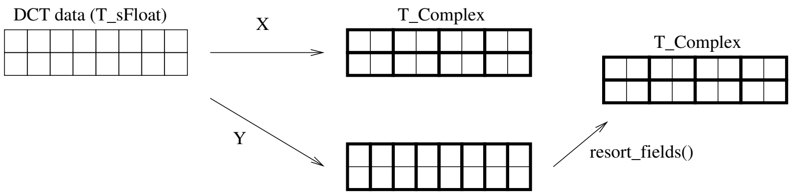
K výpočtu tabulek DCT a FFT se použijí funkce `calc_tab` pro výpočet tabulky FFT a `calc_dct_table`. Tabulka DCT obsahuje podobně jako tabulka FFT členy $W^n = e^{-\frac{i2\pi n}{N}}$, a to jeden celý kruh[8] s N body. Tedy vlastně doplňuje FFT tabulku.

Data jsou poté pomocí funkce `cp_data_and_pad` překopírována do výstupního (či pouze výpočetního prostoru). Při kopírování jsou vyplňeny konce mezi skutečnými rozměry dat a rozšířenými rozměry dat nulami.

V každém směru je pak aplikováno DCT. Nejdříve jsou vytvořena pomocná data. Tato data jsou uložena na místo původních, výpočet pomocných dat se totiž provádí inplace ve funkci `dct_make_auxdata`. Tato funkce provádí výpočet rovnice 2.24. V této funkci se poprvé využívá DCT tabulky, ovšem celá tabulka se zde musí rotovat o $\frac{\pi}{2N}$.

Poté jsou, pokud se jedná o DCT v druhé či třetí dimenzi, data přeuspořádána. Toto přeuspořádání je nutné kvůli FFT. Při výpočtu FFT je využíváno faktu, že typ `T_Complex` jsou vlastně dva typy `T_sFloat`. V první dimenzi je vše v pořádku, typy jsou správně za sebou, reálné a imaginární složky se střídají. V druhé a třetí dimenzi nastává problém (obr. 3.2).

Imaginární složka není za přímo za reálnou a musí se provést přeuspořádání dat, tak aby



Obrázek 3.2: Přeuspořádání pomocných dat DCT

tomu bylo. Toto vykonává funkce `resort_fields`. Princip se dá demonstrovat na dvou řádkách. Chceme-li dva řádky dat uložit po sloupcích, pak se index všech prvků v prvním řádku zdvojnásobí a index všech prvků v druhém řádku "přeteče" na lichá čísla, která vyplní mezery mezi prvky z prvního řádku.

Po přeuspořádání jsou data připravena na reálné FFT. Je provedena trojice operací : reorder (`reorder_and_copy`) výpočet butterfly (`calc_buttefly_in_direction`) a poslední krok reálného FFT (`fft_last_step_in_direction`). Poté jsou data opět přeuspořádána ovšem tentokrát zpět. Nyní máme FFT pomocných dat a z něj je třeba dovypočítat DCT. K tomu účelu poslouží funkce `get_dct_from_auxdata`. Tato funkce opět využívá DCT tabulku, tentokrát ji ovšem nerotuje. V této funkci se využívá pomocný paměťový prostor, jelikož je v ní obsažena rekurze, kde se musí uchovávat sumy (viz. rov. 2.28).

V případě, že jde o výpočet DCT po blocích šířky 8 použijí se optimalizované funkce `dct8` a

[8] oproti tabulce FFT, kde je potřeba jen polovina kruhu

idct8. Popis těchto funkcí zde není uveden, neboť tento algoritmus již je v značně obsáhlé formě rozebrán v teoretické části (viz. kap. 2.2.1). DCT obraz dat má díky tomu, že není periodický, ale symetrický tu vlastnost, že směrem od počátku frekvence rostou, což znamená, že na poslední pozici v obrazu je nejvyšší frekvence DCT.

3.2.3 Hartleyova transformace

Datová struktura

Podobně jako u frekvenčních dat a dat transformovaných DCT je v oblasti Hartleyovi transformace nutné znát jen rozměry dat a data samotná. Ostatní datové prvky struktury jsou opět jen přenášeny pro pozdější použití.

Typ	Název	Stručný popis
`T_sInt`	`Res[3]`	Rozměry FHT dat
`T_sFloat`	`Dis[3]`	Distorze původních dat
`T_sInt`	`Original_Res[3]`	Rozměry obrazových dat
`P_sFloat`	`Data[LATTICE_MAXLAYERS]`	Datové prostory
`T_sInt`	`Original_cell_type[LATTICE_MAXLAYERS]`	Typy dat
`T_ColorModel`	`Color_model`	Barevný model
`char`	`Caption[255]`	Název dat
`char`	`Source[255]`	Zdroj dat
`T_sInt`	`Pos[3]`	Počáteční pozice dat
`T_sInt`	`Rot[3]`	Počáteční rotace dat
`T_Int`	`Min`	Minimální hodnota v datech
`T_Int`	`Max`	Maximální hodnota v datech
`T_Int`	`StartTreshold`	Začátek oříznutí

Tabulka 3.8: Datová struktura pro data transformovaná FHT `T_FreqHart_Data`

Výpočetní náročnost

Tak jako u FFT je výpočet FHT proveden pomocí postupného půlení intervalu, jedná se tedy opět o decimaci v čase. Provede se tedy $log_2 N$ rozdělení intervalu. V každém rozdělení se spočítají všechna butterfly. Butterfly rychlé Hartleyovi transformace obsahuje 4 násobení a 6 součtů. Jedná se vlastně o jedno komplexní násobení, když zanedbáme 4 součty. Ovšem butterfly rychlé Hartleyovi transformace kombinuje 4 vstupy do 4 výstupů. Je tedy nutné v každém rozdělení provést výpočet $\frac{N}{4}$ těchto butterfly. Dostáváme tedy celkovou složitost $\mathcal{O}\left\{\frac{N}{4} log_2 N\right\}$. Tato výpočetní náročnost se vyrovná i reálnému FFT, ovšem pouze v jednodimenzionálním případě. Díky tomu, že Hartleyova transformace není separabilní a je nutno použít separabilní jádro a poté aplikovat korekční mechanismus (2.40) či (2.42), je ve vícedimenzionálním případě Hartleyova transformace pomalejší nežli reálné FFT.

Implementace rychlé Hartleyovi transformace

Implementace rychlé Hartleyovi transformace je uvedena ve zdrojových souborech `transform_hartley.c` a `transform_hartley.h`.

Řídícími funkcemi jsou v tomto případě `fht` a `ifht`. Tak jako u FFT a DCT je nejdříve zkontrolován datový typ, poté alokován výstupní deskriptor a upraveny velikosti jako u FFT. Poté jsou alokovány výstupní datové prostory. FHT používá stejnou tabulku W_N^n jako inverzní FFT. Tedy k vytvoření této tabulky použije funkci calc_tab ze souboru transform_fourier.c.

Poté se použije reorder_and_copy opět ze souboru transform_fourier.c. Postupným voláním funkce fht_butterfly_calc_in_direction pro všechny dimenze spočítám Hartleyovu transformaci nebo pomocnou separabilní Hartleyovu transformaci v případě, že se jednalo o vícedimenzionální data. Vícedimenzionální Hartleyovu transformaci musím ještě dopočítat korekčním mechanismem. Korekčním mechanismus pro dvoudimenzionální data je implementován ve funkci fht_recalc_2D, pro třídimenzionální data pak ve funkci fht_recalc_3D.

Díky "blízké příbuznosti" s DFT je uložení DHT obrazu dat obdobné jako u FFT. Počátek spektra je posunutý do počátku dat a situace s frekvencemi je totožná jako u DFT.

3.2.4 Filtry

Byly implementovány jednoduché vícedimenzionální obdélníkové filtry. Poznamenejme ze ve vícedimenzionálním případě se místo obdélníkové raději říká poloměrové, protože frekvence rostou lineárně se vzdáleností od počátku.

Ve zdrojových souborech filter_fourier.c, filter_cosine.c a filter_hartley.c jsou tedy implementovány poloměrové filtry pro jednotlivé transformace.

V každém z těchto souborů je jedna hlavní funkce, která nese název souboru (bez přípony). V každé z těchto hlavních funkcí je zajištěno, že se zkontroluje datový typ, alokuje potřebné množství paměti a provede vlastní filtrace.

Vlastní filtrace je implementována v souboru utils_filter.c. Vykonání filtrace sestává z volání dvou funkcí:

filter_prep , která analyzuje nastavení předané modulu a vrátí upravené poloměry pro filtraci, a některé další proměnné nutné ke správnému provedení filtru

filter , která provede vlastní filtraci.

Jelikož je implementován poloměrový filtr, byla vytvořena i struktura, která popisuje bod na kružnici. I když bychom měli pracovat spíše s elipsou stačí nám kružnice.

K výpočtu bodu na kružnici je používáno Pittewayova algoritmu pro generování kružnic.

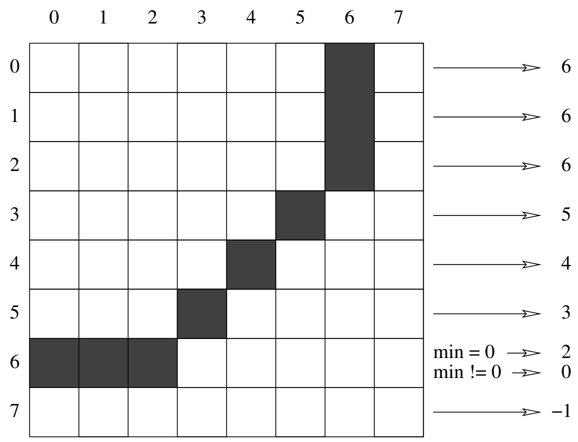
Typ	Název	Stručný popis
int	r	poloměr kružnice
int	x	x-ová souřadnice v rastru
int	y	y-ová souřadnice v rastru
int	d	koeficient Pittewayova Algoritmu
int	part	část kružnice : 0-45°,45-90°, 90°
int	min	Minimum či maximum na řádce

Tabulka 3.9: Datová struktura popisující bod na kružnici v rastrovém prostředí T_POC

Negenerujeme elipsy protože využíváme jednoduchého faktu, že rozměry v transformované oblasti jsou vždy mocniny dvou, tudíž elipsu získám pouhým vynecháním řádek z kružnice, či bitovým posunem indexu sloupce.

Ve struktuře T_POC jsou zajímavé prvky part a min. Například mějme dvoudimenzionální data, která chceme filtrovat. Vygenerovali by jsme tedy kružnice pro horní a dolní mez propusti a kde by se nám protly řádky s těmito kružnicemi, tam by byly horní a dolní meze pro filtrace řádků. Problém nastane pokud se dostaneme nad úhel 45°, kde může kružnice sledovat řádek několik bodů. V tomto případě záleží na typu meze. Jestliže je to dolní mez propusti je nutno brát jako bod, kde kružnice protla řádek, bod s nejmenším indexem (minimální), ale pokud jde o horní mez propusti je nutno brát bod s největším indexem (maximální). Kritickou částí kružnice je tedy oblast part=45-90°. Prvek min určuje jak se bude kružnice v této oblasti chovat. Celý mechanismus je rekurzivní, což znamená, že když máme třídimenzionální data, tak výpočtem horní a dolní meze klesneme o dimenzi níž tedy do jedné plochy v datech. Pak se provede to samé, dokud se nedostaneme na horní a dolní mez na řádku.

Pro filtraci v oblasti DCT po blocích 8 je používáno již vygenerovaných kružnic, takže se



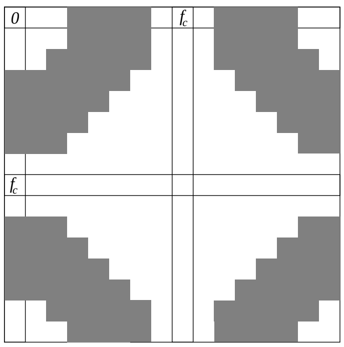
Obrázek 3.3: Význam prvku `min` ve struktuře `T_POC`

již nepoužívá Pittewayova algoritmu. Kružnice jsou uloženy ve formě polí, do kterých se pomocí poloměrů odkazuje funkce provádějící filtraci DCT8 dat pro získání horní a dolní meze propusti.

Způsoby filtrace v transformovaných oblastech

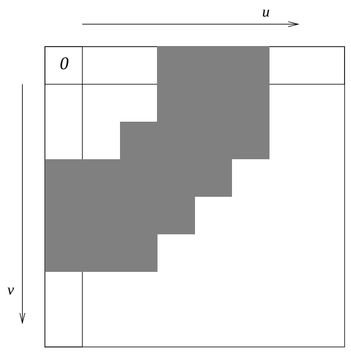
Každý obdélníkový filtr se vztahuje k počátku, k nulové frekvenci. Obdélníkový filtr ve vícedimenzionálním případě dělá okolo počátku kruhové oblasti, které má či nemá odfiltrovat. Podle způobu uložení obrazu dat se musí provádět i filtrace. Způsob filtrace se zároveň liší u jednotlivých transformací.

FFT & FHT - Díky tomu, že jsou data uložena posunutá je filtr uzpůsoben tak, že filtruje "z obou stran spektra". (viz. obr. 3.4).



Obrázek 3.4: Příklad filtrace u posunutého spektra

DCT - Díky symetrii je nutno brát celý obraz DCT jako obdobu jednoho kvadrantu (kvadrantu s nulovou frekvencí). (viz. obr. 3.5).



Obrázek 3.5: Příklad filtrace DCT obrazu dat

3.2.5 Ostatní implementované části

Aby bylo možné plně otestovat diskrétní transformace pod MVE, bylo ještě zapotřebí vytvořit některé další moduly pro MVE. Tyto moduly byly rovněž zahrnuty do knihovny `transform.dll`.

Načítání a ukládání BMP souborů

Bylo nutné vytvořit moduly, které by byly schopné načítat a ukládat obrazová data ve vhodném formátu. Byl zvolen formát Microsoft Windows BMP především pro svojí jednoduchou implementaci a značnou rozšířenost.

Implementace načítání a ukládání BMP souborů je umístěna v souboru `lattice_io.c` a v jeho hlavičkovém souboru `lattice_io.h`.

Tato implementace umožňuje

- Načítat BMP obrázky s bitovou hloubkou 1,4,8,24 bitů, obrázky s bitovou hloubkou mohou obsahovat RLE kompresi. Pro načítání BMP je v MVE určen modul `Lattice_Loader`.
- Ukládat BMP obrázky s bitovou hloubkou 24 bitů. Pro ukládání BMP je v MVE určen modul `Lattice_Saver`.

Funkce, které řídí celý proces načítání či ukládání BMP obrázků se jmenují `read_bmp_to_lattice` a `save_lattice_to_bmp`.

Při načítání nejdříve funkce `read_bmp_to_lattice` zjistí z hlavičky vstupního souboru, jehož jméno dostal v nastavení modulu, o jaký typ BMP souboru se jedná, tj. kolik je bitová hloubka, rozlišení, typ komprese apod. V případě, že se jedná o indexovanou bitmapu přečte nejdříve tabulku barev a poté spustí jednu z funkcí pro čtení indexované (`read_indexed_bmp`) či RLE komprimované (`read_rle_bmp`) bitmapy. Pokud se jedná o soubor s bitovou hloubkou 24 pak postupně přečte celou RGB bitovou mapu.

Při ukládání se nejprve zjistí, zda-li se jedná skutečně o obrázek v barevném modelu RGB. Pokud tomu tak je je tento obrázek uložen na disk.

Konverze mezi daty Lattice a Volume

Jelikož v MVE již existovala struktura popisující volumetrická data `T_Volume_Data` a pro diskrétní transformace bylo vhodné zavést obecnější strukturu `T_Lattice_Data`, bylo nutné zajistit bezpečnou konverzi mezi `T_Lattice_Data` a `T_Volume_Data` oběma směry.

Proto vznikly moduly `Lattice_2_Volume` a `Volume_2_Lattice`.

Tyto moduly jsou z části implementovány v samostatném zdrojovém souboru `convert.c` a dále ve zdrojovém souboru `transform.cpp` umístěného v podadresáři `win`.

K převodu `Volume_2_Lattice` dojde vždy, zde není převod ničím ohrožen. Ovšem při převodu `Lattice_2_Volume` se musí `Lattice_2_Data` splňovat tyto podmínky:

1. Barevný model `Color_model` musí být V
2. Data musí být třídimensionální

Struktury s nastavením, MVE dialogy

Nastavení modulu (`CUSTOM_MODUE_DATA`) je předáváno modulu společně s datovými descriptory (`DATA_DESCRIPTORS`). Nastavení diskrétních transformací se provádí pouze u transformací dopředných, inverzní transformace se nenastavují. Všechny transformace mají velmi podobné struktury s nastavením.

Nastavení každé transformace obsahuje `layer_mask`, bitovou masku, kde uživatel nastavuje barevné složky které se mají transformovat, ostatní složky se "zahodí".

V nastavení rychlé Fourierovi transformace je možné zvolit zda se má provádět pomalejší transformace přes normální FFT parametrem `force_cmplx`.

U DCT je možné nastavit zda se provede DCT po blocích velikosti 8 či po přes celá data a to pomocí prvku `dct8`.

Každá datová struktura s nastavením transformace je definována v hlavičkovém souboru každé transformace.

Dále bylo potřeba nastavení pro filtrace. Filtrace ve všech oblastech transformací sdílejí

Typ	Název	Stručný popis
`int`	`layer_mask`	bitová maska určující složky
`int`	`force_cmplx`	

Tabulka 3.10: Struktura s nastavením ryché Fourierovi transformace `FFTSETTINGS`

Typ	Název	Stručný popis
`int`	`layer_mask`	bitová maska určující složky
`int`	`dct8`	příznak dct8

Tabulka 3.11: Struktura s nastavením diskrétní cosinové transformace `DCTTSETTINGS`

Typ	Název	Stručný popis
`int`	`layer_mask`	bitová maska určující složky

Tabulka 3.12: Struktura s nastavením rychlé Hartleyovi transformace `FHTSETTINGS`

jeden typ struktury. Datovou strukrurou s nastavením filtrů je `FILSETTINGS`. Tato struktura je definována v hlavičkovém souboru `utils_filter.h`.

Typ	Název	Stručný popis
`int`	`type`	typ filtru (zatím nevyužito)
`int`	`bounds[2]`	dolní a horní mez filtrace

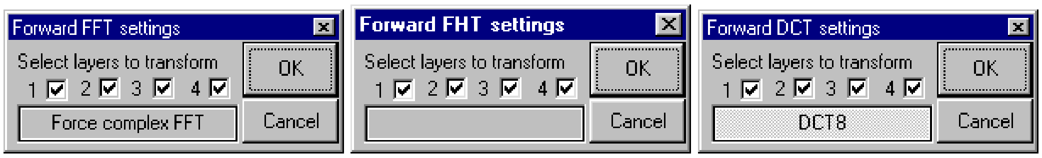
Tabulka 3.13: Struktura s nastavením filtru `FILSETTINGS`

ke každému nastavení musel být vytvořeno dialogové okno v prostředí Microsoft Windows. Byly použity standartní prostředky, které poskytuje Microsoft Visual C++ 6.0. Dialogy byly implementovány pouze dva: jeden dialog pro transformace, druhý dialog pro filtry. Dialogy jsou implementovány v souborech `win/DialogTransformSetup.cpp` a `win/DialogFilterSetup.cpp`.

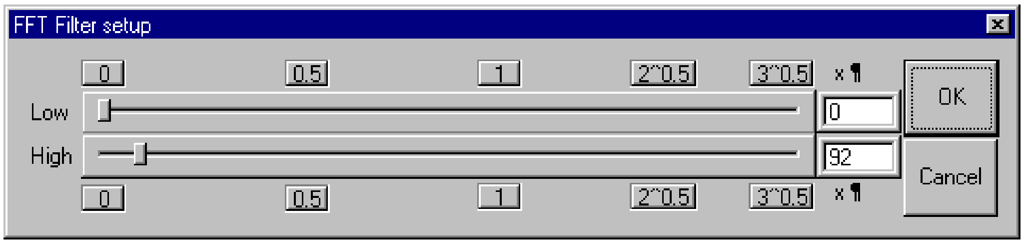
Tyto dialogy jsou volány z knihovny `transform.dll`, v níž jsou implementovány a dle typu transformace se mění jejich vzhled a zacházení s jejich proměnnými. Kromě ostatních prvků objektů má totiž každý dialog public proměnné, které jsou z knihovny změněny podle typu transformace a dřívějšího nastavení.

U transformací jsou to celočíslené proměnné `layermask`, `various` a `checkstate` a dále řetězce `caption` a `checkcaption`. Oba řetězce a `checkstate` nastavují vzhled dialogu. Proměnné `layermask` a `various` slouží k ukládání stavu nastavení uvnitř dialogu a toto nastavení je při skončení dialogu přečeto knihovnou a nastavení je změněno.

U filtrací je to celočíselné pole `bounds` a řetězec `caption`. Řetězec `caption` mění titulek okna a celočíslené pole `bounds` slouží k uchování nastavení dialogu.



Obrázek 3.6: Dialogy nastavení diskrétních transformací



Obrázek 3.7: Dialog nastavení filtrace

3.2.6 Začlenění do jediné knihovny

Všechny moduly bylo potřeba začelnit do jediné `dll` knihovny, a vytvořit rozhraní komunikace mezi MVE a touto knihovnou. Tato část implementace je v souboru a `win/transform.cpp` a částečně též v souboru `win/transform.h`. Komunikace mezi MVE a moduly je dle potřeby přeměřována či upravována a kontrolována. Zároveň je zde implementována obsluha dialogových oken s nastavením.

3.3 Zhodnocení výsledků

Diskrétní transformace byly testovány na platformě Windows NT, na počítači Intergraph, Intel Pentium II 400Mhz, 512 MB RAM. Jednotlivé transformace byly testovány na netriviálních datech. V tabulkách 3.14, 3.15,3.16 a 3.17 jsou uvedeny půměrné časy výpočtu transformací. Hodnoty jsou uvedeny ve tvaru sekundy:milisekundy.

Výpočetní čas filtrací je vzhledem k transformacím zanedbatelně malý, tudíž jej zde neuvádím.

Obrázek 3.8: Testované obrázky `Beach` a `Suculent`

směr	FFT	reálné FFT	DCT	DCT8	FHT
dopředná	3:312	1:593	2:984	0:079	2:734
inverzní	3:390	1:796	3:203	0:188	2:891

Tabulka 3.14: Výpočetní čas jednotlivých metod dvoudimenzionálních (obrazových) dat : `Beach.bmp`, 312x400 pixelů, RGB

směr	FFT	reálné FFT	DCT	DCT8	FHT
dopředná	7:865	3:609	6:500	0:203	6:250
inverzní	7:531	3:969	6:922	0:547	6:531

Tabulka 3.15: Výpočetní čas jednotlivých metod dvoudimenzionálních (obrazových) dat : `Suculent.bmp`, 680x460 pixelů, RGB

Obrázek 3.9: Testovaná volumetrická data `Syn-64` a `Ctmayo`

směr	FFT	reálné FFT	DCT	DCT8	FHT
dopředná	0:781	0:359	1:078	0:094	0:562
inverzní	0:890	0:437	1:125	0:187	0:625

Tabulka 3.16: Výpočetní čas jednotlivých metod třídimenzionálních (volumetrických) dat : `syn-64.vol`, 61x61x50 voxelů, V

směr	FFT	reálné FFT	DCT	DCT8	FHT
dopředná	7:688	3:672	9:531	0:797	5:765
inverzní	8:438	4:281	10:141	1:640	6:375

Tabulka 3.17: Výpočetní čas jednotlivých metod třídimenzionálních (volumetrických) dat : `ctmayo.vol`, 108x109x97 voxelů, V

3.3.1 Zhodnocení jednotlivých transformací

Frekvenční oblast (FFT)

Rychlá Fourierova transformace je nejvhodnější transformací ke zpracování obrazu. Má velmi rychlé a jednoduché implementace. Z tabulek 3.14, 3.15,3.16 a 3.17 je vidět, že reálné FFT se ukázalo jako nejrychlejší transformací z implementovaných transformací, nepočítáme-li dct po blocích 8.

Značnou výhodou reálného FFT budiž také téměř poloviční úspora paměti proti komplexnímu FFT.

Oblast Hartleyovi transformace (FHT)

Hartleyova transformace v první dimenzi má stejnou rychlost jako reálné FFT (nebylo testováno). Ovšem nevýhodou Hartleyovi transformace je nutnost korekčního mechanismu ve vícedimenzionálním případě. Díky tomuto zpomalení je Hartleyova transformace méně atraktivní ve vícedimenzionálním případě, nežli reálná FFT. Rychlost Hartleyovi transformace ve vícedimenzionálním případě se pohybuje v polovině mezi reálným FFT a normálním komplexním FFT.

Výsledky filtrací v Hartleyově oblasti jsou stejné jako u filtrací skrze FFT.

Oblast diskrétní cosinové transformace (DCT)

U DCT jsme očekávali poněkud lepší výpočetní čas. Ze všech trasnformací má nejhorší výsledky ohledně rychlosti transformace. Je něklik důvodů proč je zpomalení tak velké:

1. Tato implementace DCT vychází z výpočtu rovnice 2.20 ovšem v této rovnici odsune činitel C_n a provádí tak de facto jen poloviční transformaci. V našem případě ovšem potřebujeme DCT tak jak má být, tedy i s tímto činitelem proto je nutné celá data násobit navíc C_n.

2. Předchozí bod je potřeba udělat v každém směru proto se vzrůstajícím počtem dimenzí výpočetní čas dále narůstá.

Oproti tomu je DCT8 velmi rychlá, i když vlastně musíme provádět jednu část kvantizace (násobení C_n).

DCT má jednu velkou výhodu, která z ní dělá velmi atraktivní transformaci pro kompresi obrazu. DCT reprezentace dat má totiž na začátku intervalu obsaženo ještě více informace nežli FFT a potažmo pak FHT.

Kapitola 4

Závěr

Jednotlivé transformace, filtry a další moduly byly implementovány úspěšně do prostředí MVE. Testováním se zjistilo že nejrychlejší implementovanou transformací je reálné FFT , které má zároveň menší paměťové nároky nežli normální komplexní FFT. FHT je poněkud pomalejší nežli reálné FFT, ale má ještě menší paměťové nároky. Diskrétní cosinová transformace se ukázala být velmi pomalou, což je způsobeno nutností dalších násobení po samotném výpočtu transformace. Diskrétní cosinová transformace po blocích velikosti 8 je velmi zdařile implemnetována, ovšem její použití jí odkazuje více na kompresi obrazu, nežli ke zpracování vyšší úrovně.

Filtrace jsou velmi rychlé díky rychlému výpočtu oblastí filtrů a byly otestovány na obrazových i volumetrických datech. V příloze jsou umístěny některé výsledky.

Literatura

[1] Strang G. : *The Discrete Cosine Transform* http://www-math.mit.edu/ gs

[2] Teukolsky,S.,A., Vetterling, W.,T., Flanney, B.,P., *Numerical Recipes in C: The Art of Scientific Computing* Second Edition, Cambridge Univerzity Press, 1996 , http://www.nr.com

[3] Anonymised paper : *The Multi-Dimensional Hartley Transform as a Basis for Volume Rendering*

[4] Watson, B. Poirson, A. : *Separable two-dimensional discrete Hartley transform J.Opt.Soc., Am.A.*

[5] Ullman,R.,F. : *An algortihm for fast Hartley Transform* paper

[6] Pennebaker, W.,B., Mitchell, J.,L., : *Jpeg still image data compression standart*, Van Nostrand Reinhold, New York 1993 ISBN 0-442-01272-1

[7] Skala, V. : *Počítačová grafika 1 - 3*, 2 vydání VŠSE v Plzni, 1991, strany 40,41

Příloha A

Filtrace ve Fourierove prostoru