En introduktion till krusningar via Haarsystemet

Lennart Jern, 34305
Avhandling pro gradu

Fakulteten för naturvetenskaper och teknik
Matematik
Åbo Akademi

20 september 2016

Sammanfattning

Krusningar (*eng.* wavelets) är en sorts basfunktioner som används t.ex. inom signalbehandling och komprimering. Den mest grundläggande av alla krusningar är Haarkrusningen och den används i den här avhandlingen som utgångspunkt för tillämpningar och exempel i form av bildkomprimering.

Målet har varit att undersöka hur krusningar kan användas i praktiken och bygga upp en intuition för hur Haartransformen fungerar. Stöd för förståelsen ges med hjälp av beskrivande figurer och visuella exempel på komprimeringens effekter.

Utgående från en krusning går det alltid att bilda en familj av krusningar som tillsammans utgör en ortonormerad bas för Hilbertrummet $L^2(\mathbb{R})$. Med hjälp av en sådan bas går det sedan att dela upp en signal i olika beståndsdelar för analys eller modifiering. Exempelvis kan en signal som innehåller brus på en hög frekvens delas upp i höga och låga frekvenser. På så sätt går det att göra sig av med bruset om krusningsbasen är väl vald.

Krusningstransformen kan ses som en generalisering av Fouriertransformen med den skillnaden att krusningarna ger en tidslokal frekvensanalys. Det här är speciellt viktigt för behandling av signaler med flera tidslokala delar. I ett musikstycke kan t.ex. varje takt vara uppbyggd kring ett skilt ackord och då är det viktigt att kunna analysera signalen tidslokalt för att inte blanda ihop alla ackorden.

Bildkomprimering har blivit oerhört viktigt i och med digitalkamerornas utveckling och introduktionen av telefoner med inbyggd kamera. Digitalkamerorna har fått högre upplösning och har nu ofta mellan 10 och 20 miljoner pixlar per bild. Det produceras också allt större mängder bilder som sedan dessutom sprids flitigare än förr med hjälp av olika tjänster på internet. Komprimeringen behövs dels för att stora mängder bilder praktiskt ska kunna lagras och dels för att bilderna också ska kunna sändas snabbt över internet.

Krusningarna lämpar sig väl för bildbehandling och -komprimering eftersom två punkter nära varandra i en bild ofta har någonting gemensamt (t.ex. samma färg eller ljusstyrka). Om bilden tolkas som en signal ligger de här punkterna nära varandra i tiden istället och det finns därför orsak att analysera bilden tidslokalt. Implementationen av bildkomprimeringen i den här avhandlingen är gjord i Octave med en in situ-algoritm.

Haarkrusningen fungerar bra som introduktion till krusningar, men den lämpar sig inte för bildkomprimering annat än i vissa specialfall. Däremot ger Haarsystemet enkla beräkningar som går bra att göra också för hand i mindre skala. Haarsystemet ger också en naturlig koppling till de enkla funktioner som används för att bygga upp den allmänna teorin kring krusningar.

Innehåll

Inledning

Alfréd Haar (1885-1933) var en ungersk matematiker som studerade för David Hilbert och som i sin doktorsavhandling undersökte ortonormerade system av funktioner på intervallet $[0, 1]$. I avhandlingen introducerade han det som vi idag kallar Haarkrusningen och Haarsystemet.

Traditionellt härleds krusningarna ofta ur fouriertransformen genom konstruktion av en tidslokal bas. En tidslokal vektor har ett litet område med komponenter olika 0 medan resten av vektorns komponenter är 0 eller åtminstone nära 0. I t.ex. bildbehandling gäller det ofta att ett område av bilden inte har någon koppling till ett annat område och då passar en tidslokal bas bra för att dekomponera bilden. Om bilden uttrycks med hjälp av basen på följande sätt $\sum_{i=0}^{m} k_i \cdot w_i$, där w_i är de olika basvektorerna och k_i är konstanter, fungerar varje term som ett skikt i bilden. Eftersom basvektorerna är tidslokala påverkar varje term bara en liten del av bilden och det går därför att ta bort termer ur bilden utan att förstöra helheten.

Mitt mål med den här avhandlingen har varit att undersöka hur krusningar kan användas i praktiken och bygga upp en intuition för vad Haartransformen går ut på. Jag har strävat till att på ett lättförståeligt sätt förklara vad krusningar är och hur de kan användas. Genom att använda Haarkrusningarna som exempel och tillämpa teorin på bildkomprimering har jag försökt skapa en stark verklighetsförankring och ge en intuitiv känsla för hur krusningarna fungerar. Jag har lagt ner mycket tid på att konstruera figurer som komplement till texten och min förhoppning är att de ska kunna ge en snabb översikt över avhandlingens innehåll samt bidra till en angenäm läsupplevelse.

Avhandlingens upplägg

I det första kapitlet introduceras krusningsbegreppet gradvis utgående från Haarkrusningen och stegfunktioner över dyadiska intervall. Mot slutet av kapitlet introduceras approximationsrummet V_j, bestående av alla stegfunktioner med skalan j. Elementen i V_j kan användas för att approximera funk-

tioner ur Hilbertrummet $L^2(\mathbb{R})$.

Haarsystemets skalningsfunktioner utgör en bas för V_j medan krusningarna utgör bas för W_j som består av alla $f \in V_{j+1}$ sådana att $\langle f, g \rangle = 0$, $\forall g \in V_j$. Med hjälp av krusningarna är det alltså möjlig att göra en ortogonal uppdelning av vektorrummet $V_{j+1} = V_j \oplus W_j$. Jag har i det här kapitlet främst utgått från [1] och [2], men också [3], [4], [5] och [6] har hjälpt mig på vägen.

I det andra kapitlet presenteras tolkningen av digitala bilder som signaler. Signalerna kan delas upp i olika beståndsdelar med hjälp av en krusningstransform för vidare behandling eller komprimering. Eftersom beräkningarna snabbt blir tröttsamma att göra för hand övergår jag, efter några exempel, till att utföra transformen i Octave med hjälp av en in situ-algoritm. Det här gör det samtidigt möjligt att använda riktiga bilder i exemplen. I det här kapitlet har jag haft stor hjälp av [3], speciellt vid implementationen i Octave.

Det tredje kapitlet behandlar komprimering med hjälp av krusningar. Till en början används svartvita foton för enkelhetens skull, men senare i kapitlet visas också exempel på hur komprimeringen kan göras med färgbilder. I slutet av kapitlet visar jag hur man med hjälp av matrisnormer kan mäta hur stor skada komprimeringen orsakar på bilden. Det här är viktigt för att kunna automatisera processen utan att riskera alltför hård komprimering. I det här kapitlet har jag utgått främst från [3] vid implementeringen av komprimeringen. Dessutom har jag haft nytta av [7] i anslutning till matrisnormerna.

I det fjärde och sista kapitlet presenteras den allmänna teorin kring krusningar. Samtidigt visas att Haarkrusningen mycket riktigt är en krusning och att Haarsystemet sålunda utgör en ortonormerad bas i Hilbertrummet $L^2(\mathbb{R})$. I det här kapitlet stöder jag mig på [8] för teori kring måtteori samt bevisen till några av satserna. Viss inspiration har jag även fått från [9].

I tillämpningarna och exemplen har jag genom hela avhandlingen till stor del använt mig av Octave, som är ett högnivåspråk ämnat för numeriska beräkningar, och därmed har jag ofta också haft nytta av [10]. All programkod som behövs för att testa exemplen finns i bilaga A i slutet av avhandlingen.

För sammanställningen av den här inledningen har jag fått inspiration från [11] och [12].

Strukturerade härledningar

Bevis och beräkningar av olika slag presenteras i form av strukturerade härledningar (*eng.* structured derivations) [13]. Jag har valt att använda mig av strukturerade härledningar eftersom jag tycker att de på ett tydligt och

logiskt sätt visar hur bevisen är uppbyggda. De strukturerade härledningarna gör det lätt att se vilka antaganden som görs och vilka delar ett bevis består av.

Jag har valt att använda beteckningen $(a), (b), \ldots$ för antaganden och $(1), (2), \ldots$ för observationer för att inte blanda ihop dem med bibliografiska referenser. Beteckningarna här är alltså inte riktigt samma som i [13]. Här följer en kort förklaring över de olika delarna i en strukturerad härledning. För en mera ingående beskrivning se [13].

- Här beskrivs uppgiften: Vad ska göras?

(a) Det här är första antagandet.

(b) Det här är andra antagandet.

(1) {Här är förklaringen till att den första observationen gäller.}

Det här är första observationen.

$\Vdash$ {Motivering till att uträkningen löser uppgiften (om det behövs).}

Här börjar själva uträkningen.

rel {Här ges en motivering till följande steg.}

Här är följande steg.

Kapitel 1

Haarsystemet

1.1 Vad är en krusning?

Vardagligt sagt är en krusning en liten (kort) våg eller en kort oscillation. Det handlar alltså inte om en begränsning av amplituden utan om vågens utbredning i tiden. En mer matematisk beskrivning kunde kanske se ut så här: En krusning är en funktion som är konstant 0 förutom i ett begränsat område. I praktiken används krusningarna inom t.ex. signalbehandling för att analysera signaler (data) genom att dela upp dem i komponenter. Uppdelningen kan anpassas både till indata och ändamål vilket gör krusningarna väldigt användbara. Komponentuppdelningen görs med hjälp av en krusningsbas i vektorrummet för signalerna (ofta $L^2(\mathbb{R})$) och basen konstrueras utgående från en fader- och en moderkrusning. Faderkrusningen kallas även för skalningsfunktion.

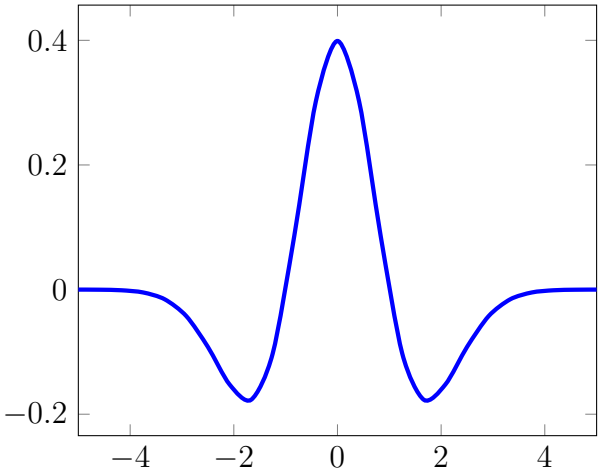
1.2 Haarkrusningen

Redan innan den allmänna teorin för krusningar utvecklades var Haarkrusningen väl känd. Den är den mest grundläggande av alla krusningar och utgör moderkrusningen för Haarbasen.

Definition 1.1. Haarkrusningen definieras för $x \in \mathbb{R}$ som

$$H(x) = \begin{cases} 1 & \text{, då } x \in [0, \frac{1}{2}) \\ -1 & \text{, då } x \in [\frac{1}{2}, 1) \\ 0 & \text{, annars} \end{cases}$$

Den del av krusningen som är olika 0 finns i ett så kallat *dyadiskt intervall*, dessutom är hela $H(x)$ en *dyadisk stegfunktion*.



Figur 1.1: En krusning formad som en mexikansk hatt.

Definition 1.2. Ett *dyadiskt intervall* är ett intervall av typen

$$I_{j,k} = [2^{-j}k, 2^{-j}(k+1))$$

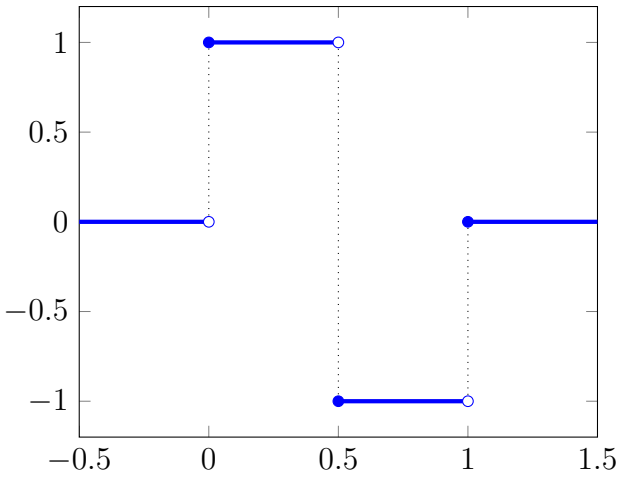
med $j, k \in \mathbb{Z}$. En funktion som är konstant på varje intervall $I_{j,k}$ för ett givet j kallas för en *dyadisk stegfunktion* med skalan j.

Det dyadiska intervallet bestäms alltså av parametrarna j och k så att j kontrollerar längden på intervallet och k positionen på tallinjen. För ett fixt j blir längden $2^{-j}(k+1) - 2^{-j}k = 2^{-j}$ vilket betyder att en dyadisk stegfunktion består av konstanta ”trappsteg” av längden 2^{-j} (se figur 1.3). Märk att då j ökas blir intervallen kortare eftersom längden ges av 2^{-j}.

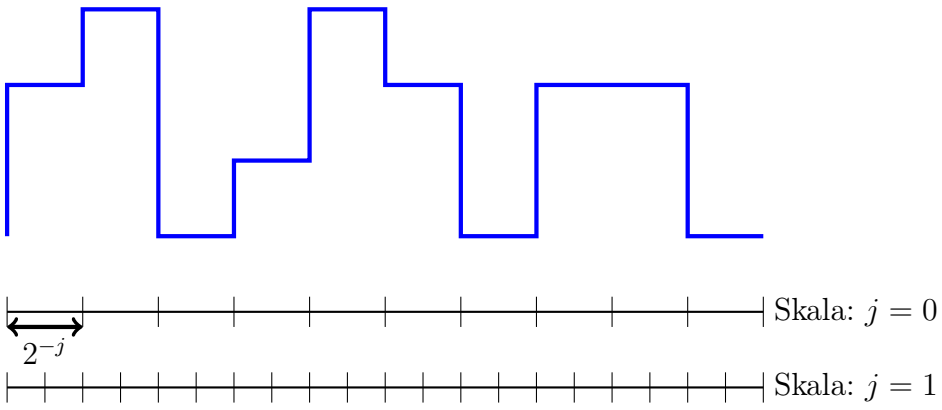
Det finns flera olika typer av data som kan tolkas som eller formas om till dyadiska stegfunktioner. En ljudsignal kan lagras i form av en vektor där varje värde anger amplituden på ljudvågen. Det är då underförstått att värdena har lagrats med en viss samplingsfrekvens som anger hur lång tid det är mellan de olika värdena. En digital bild kan t.ex. representeras av en matris med värden som anger vilken färg varje pixel ska ha. Varje rad i matrisen kan då tolkas som en dyadisk stegfunktion med steglängden samma som längden av en pixel.

1.3 Haarbasens skalningsfunktioner

Som tidigare nämnts behövs det också en skalningsfunktion för att konstruera en krusningsbas.



Figur 1.2: Haarkrusningen



Figur 1.3: Ett exempel på en dyadisk stegfunktion och dyadiska intervall för två olika värden på j.

Definition 1.3. Haarbasens *skalningsfunktioner* av ordning j definieras som

$$p_{j,k}(x) = 2^{\frac{j}{2}} p(2^j x - k)$$

där $p(x)$ ges av

$$p(x) = \begin{cases} 1 & \text{, då } 0 \leq x < 1 \\ 0 & \text{, annars} \end{cases}$$

och $j, k \in \mathbb{Z}$.

Definition 1.4. Låt $f : \mathbb{R} \mapsto \mathbb{R}$ och $M = \{x \in \mathbb{R} : f(x) \neq 0\}$. *Stödet* för f ges då av $\overline{M}$ d.v.s. det slutna höljet av det område där f är olika 0.

På motsvarande sätt som för de dyadiska intervallen bestämmer j längden och k positionen för $p_{j,k}(x)$:s stöd. Kopplingen till de dyadiska intervallen ger även upphov till en alternativ definition av skalningsfunktionerna.

Lemma 1.5. *Det går också att definiera* $p_{j,k}(x)$ *som*

$$p_{j,k}(x) = \begin{cases} 2^{\frac{j}{2}} & \text{, } \textit{då } x \in I_{j,k} \\ 0 & \text{, } \textit{annars.} \end{cases}$$

Bevis. Det räcker med att visa att $p_{j,k}(x) \neq 0$ är ekvivalent med att $x \in I_{j,k}$.

- $p_{j,k}(x) \neq 0$

$\Leftrightarrow$ {Ekvationen $p_{j,k} \neq 0$ gäller endast då $p(x) = 1$.}

$0 \leq 2^j x - k < 1$

$\Leftrightarrow$ {Lös ut x.}

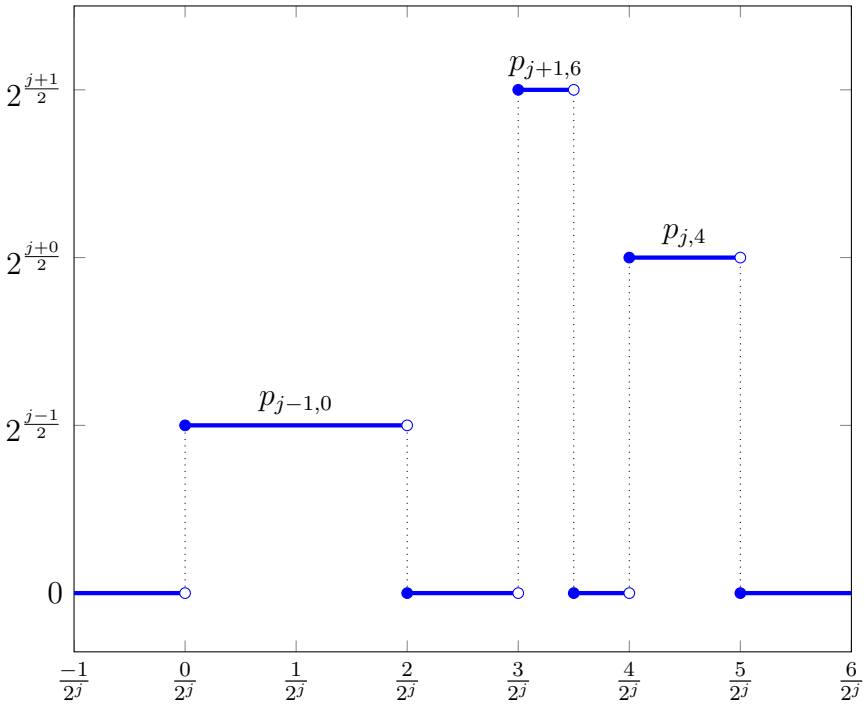
$\frac{k}{2^j} \leq x < \frac{k+1}{2^j}$

$\Leftrightarrow$ {Använd definition 1.2.}

$x \in I_{j,k}$ □

För ett givet j utgör varje skalningsfunktion en dyadisk stegfunktion med just skalan j eftersom $p(x)$ är olika 0 precis då $x \in [\frac{k}{2^j}, \frac{k+1}{2^j})$. Dessutom är $p_{j,k}(x)$ en stegfunktion också för varje finare skala eftersom en funktion som är konstant på $I_{j,k}$ för alla k även är konstant på $I_{j+1,k}$.

Lemma 1.6. *Varje dyadiskt intervall* $I_{j+1,k}$ *är helt inneslutet i ett dyadisk intervall med lägre skala* $I_{j,h}$*, där* $h = \lfloor \frac{k}{2} \rfloor$.



Figur 1.4: Tre exempel på skalningsfunktioner för Haarkrusningen. Notera att k ger positionen på tallinjen i antalet dyadiska intervall från origo och det antalet är direkt beroende av j. Därför placeras $p_{j+1,6}$ vid $\frac{3}{2^j}$ på x-axeln, ty $\frac{6}{2^{j+1}} = \frac{3}{2^j}$.

Bevis.

- $x \in I_{j+1,k}$

$\Leftrightarrow$ {Använd definition 1.2.}

$\frac{k}{2^{j+1}} \leq x < \frac{k+1}{2^{j+1}}$

$\Leftrightarrow$ {Bryt ut $\frac{1}{2}$ och gör substitutionen $c = \frac{1}{2} \cdot k$.}

$\frac{c}{2^j} \leq x < \frac{c+\frac{1}{2}}{2^j}$

$\Leftrightarrow$ $\left\{ \begin{array}{l} \text{Låt } h = \lfloor c \rfloor \text{ beteckna helatalsdelen av } c \text{ och } r = (k \mod 2) \cdot \frac{1}{2}. \\ \text{Då gäller } c = h + r. \end{array} \right\}$

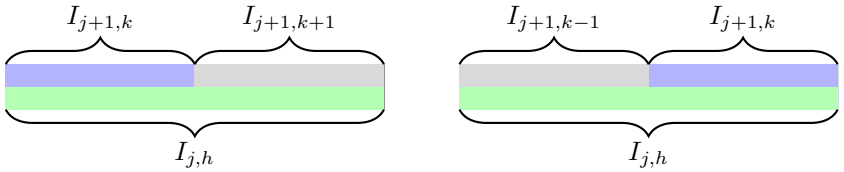
$\frac{h+r}{2^j} \leq x < \frac{h+r+\frac{1}{2}}{2^j}$

$\Leftrightarrow$ {Om k är udda är $r = \frac{1}{2}$ och om k är jämnt är $r = 0$.}

$\begin{cases} \frac{h+\frac{1}{2}}{2^j} \leq x < \frac{h+1}{2^j} & , k \text{ udda} \\ \frac{h}{2^j} \leq x < \frac{h+\frac{1}{2}}{2^j} & , k \text{ jämnt} \end{cases}$

$\Rightarrow$ {Använd definition 1.2.}

$x \in I_{j,h}$ □



Jämnt k Udda k

Figur 1.5: Beroende på om k är udda eller jämnt hamnar $I_{j,k}$ i den vänstra eller högra halvan av $I_{j,h}$

Eftersom ett intervall med skalan j är dubbelt så långt som ett med skalan $j+1$ ändrar positionerna på tallinjen så att $I_{j+1,2k}$ återfinns i $I_{j,k}$. Noteras kan också att de kortare intervallen aldrig hamnar mitt i de längre, utan alltid i den ena eller andra halvan (fig. 1.5).

1.4 Approximationsrummet V_j

Med hjälp av skalningsfunktionerna går det att bygga upp ortogonala baser för vektorrummen som består av alla dyadiska stegfunktioner med en viss skala, men först behövs några definitioner.

För en mer ingående behandling av de här begreppen rekommenderar jag [4], [5] och [6].

Definition 1.7. *Hilbertrummet* $L^2(\mathbb{R})$ består av alla funktioner f som är mätbara och som uppfyller villkoret $\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty$.

$$L^2(\mathbb{R}) = \left\{ f : \mathbb{R} \mapsto \mathbb{R} : f \text{ mätbar och } \int_{-\infty}^{\infty} |f(x)|^2 dx < \infty \right\}$$

Definition 1.8. *Inre produkten* i $L^2(\mathbb{R})$ definieras som

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x)g(x)dx,$$

för $f, g \in L^2(\mathbb{R})$.

Definition 1.9. Funktionerna f och g i ett Hilbertrum $(H, \langle , \rangle)$ är *ortogonala* om

$$\langle f, g \rangle = 0.$$

Den *inducerade normen* definieras som

$$\|f\| = \sqrt{\langle f, f \rangle}$$

och $f \in H$ kallas normerad om $\|f\| = 1$.

Följande allmänna resultat gäller för alla Hilbertrum. För bevis se [6].

Sats 1.10. *Låt H vara ett Hilbertrum och $\{e_n\}_{n=-\infty}^{\infty}$ en ortonormerad följd i H. Då är följande påståenden ekvivalenta:*

(a) $\{e_n\}_{n=-\infty}^{\infty}$ *är en ortonormerad bas i* H, *dvs.* $\langle x, e_n \rangle = 0 \quad \forall n \in \mathbb{Z} \Leftrightarrow x = 0$

(b) $\overline{span\{e_n : n \in \mathbb{Z}\}} = H$

(c) Varje $x \in H$ *kan skrivas som* $x = \sum_{n=-\infty}^{\infty} \langle x, e_n \rangle e_n$

(d) För $x \in H$ *gäller* $\|x\|^2 = \sum_{n=-\infty}^{\infty} |\langle x, e_n \rangle|^2$.

Låt V_j beteckna mängden av alla dyadiska stegfunktioner $f \in L^2(\mathbb{R})$ med skalan j försedd med inreprodukten ur $L^2(\mathbb{R})$. Med andra ord har vi $V_j = \{f \in L^2(\mathbb{R}) : f(x) = \sum_{k=-\infty}^{\infty} a_k p(2^j x - k)\}$. Följaktligen är V_j slutet i $L^2(\mathbb{R})$ och därmed ett Hilbertrum med inreprodukten ur $L^2(\mathbb{R})$. För ett godtyckligt j gäller då följande sats:

Sats 1.11. *Funktionerna $\{p_{j,k}\}_{k=-\infty}^{\infty}$ utgör en ortonormerad bas för V_j.*

Bevis. För att $p_{j,k}$ ska bilda en ortogonalbas krävs det enligt sats 1.10 att $\langle p_{j,k}, p_{j,m}\rangle = 0$ för $k \neq m$ och att $(\langle p_{j,k}, f\rangle = 0 \quad \forall k \in \mathbb{N}, f \in V_j) \Leftrightarrow f = 0$. Utöver det här krävs det av en ortonormerad bas att $\langle p_{j,k}, p_{j,k}\rangle = 1$. I beviset går det att göra antagandet att $k < m$ eftersom det omvända fallet följer helt analogt.

- Visa att $\langle p_{j,k}, p_{j,m}\rangle = 0$.

(a) $k \neq m$

(b) $k < m$

$\Vdash$ $\langle p_{j,k}, p_{j,m}\rangle$

$=$ {Använd inreproduktens definition.}

$$\int_{-\infty}^{\infty} p_{j,k}(x)p_{j,m}(x)dx$$

$=$ $\left\{ \begin{array}{l} \text{Dela upp integralen enligt de områden där funktionerna är} \\ \text{positiva. Enligt } (b) \text{ gäller } k < m. \end{array} \right\}$

$$\int_{-\infty}^{\frac{k}{2^j}} p_{j,k}(x)p_{j,m}(x)dx + \int_{\frac{k}{2^j}}^{\frac{k+1}{2^j}} p_{j,k}(x)p_{j,m}(x)dx + \int_{\frac{k+1}{2^j}}^{\frac{m}{2^j}} p_{j,k}(x)p_{j,m}(x)dx + \int_{\frac{m}{2^j}}^{\frac{m+1}{2^j}} p_{j,k}(x)p_{j,m}(x)dx + \int_{\frac{m+1}{2^j}}^{\infty} p_{j,k}(x)p_{j,m}(x)dx$$

$=$ {Sätt in värden.}

$$\int_{-\infty}^{\frac{k}{2^j}} 0dx + \int_{\frac{k}{2^j}}^{\frac{k+1}{2^j}} 2^{\frac{j}{2}} \cdot 0dx + \int_{\frac{k+1}{2^j}}^{\frac{m}{2^j}} 0dx + \int_{\frac{m}{2^j}}^{\frac{m+1}{2^j}} 0 \cdot 2^{\frac{j}{2}} dx + \int_{\frac{m+1}{2^j}}^{\infty} 0dx$$

$=$ {Integralen över 0 är 0.}

$$0$$

■

Det gäller alltså att $p_{j,k} \perp p_{j,m}$ för $k \neq m$.

- Visa att $\langle p_{j,k}, f \rangle = 0 \Leftrightarrow f = 0$.

(a) $f \in V_j$

$\Vdash$ $\langle p_{j,k}, f \rangle = 0$

$\Leftrightarrow$ {Använd definitionen på inre produkt.}

$$\int_{-\infty}^{\infty} p_{j,k}(x) f(x) dx = 0$$

$\Leftrightarrow$ {Dela upp integralen och stryk de termer som blir 0.}

$$\int_{\frac{k}{2^j}}^{\frac{k+1}{2^j}} 2^{\frac{j}{2}} \cdot f(x) dx = 0$$

$\Leftrightarrow$ $\left\{ \begin{array}{l} \text{Antagande } (a) \text{ innebär att } f \text{ är konstant på integrationsinter-} \\ \text{vallet och därför måste } f(x) = 0 \text{ för } x \in [\frac{k}{2^j}, \frac{k+1}{2^j}]. \end{array} \right\}$

$f(x) = 0, x \in [\frac{k}{2^j}, \frac{k+1}{2^j}]$

$\Leftrightarrow$ $\left\{ \begin{array}{l} \text{Eftersom ekvationen ska gälla för varje } k \in \mathbb{Z} \text{ måste } f(x) = 0 \\ \text{för alla } x \in \mathbb{R}. \end{array} \right\}$

$f(x) = 0, x \in \mathbb{R}$

■

Nu återstår ännu att visa att basen är normerad.

- Visa att $\langle p_{j,k}, p_{j,k} \rangle = 1$.

$\Vdash$ $\langle p_{j,k}, p_{j,k} \rangle$

$=$ {Använd inreproduktens definition.}

$$\int_{-\infty}^{\infty} p_{j,k}(x) p_{j,k}(x) dx$$

$=$ {Dela upp integralen och förenkla.}

$$\int_{\frac{k}{2^j}}^{\frac{k+1}{2^j}} 2^{\frac{j}{2}} \cdot 2^{\frac{j}{2}} dx$$

$=$ {Förenkla och utför integralen.}

$2^j \cdot (\frac{k+1}{2^j} - \frac{k}{2^j})$

$=$ {Förenkla uttrycket.}

1

■ □

Vekorrummen som uppstår då j varierar är inneslutna i varandra eftersom de dyadiska intervallen, där funktionerna i V_j är konstanta, är inneslutna i varandra enligt lemma 1.6.

$$\ldots V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \ldots$$

De här rummen kan kallas för *approximationsrum* eftersom en funktion $f \in L^2(\mathbb{R})$ kan approximeras med hjälp av funktionerna i V_j. Skalan bestämmer då hur noggrann approximationen blir eftersom steglängden på funktionerna i V_j är direkt beroende av j. Genom att låta j variera fås en mängd approximationsrum med godtyckligt fin eller grov skala vilket kan vara av nytta vid analys av olika delar av en signal.

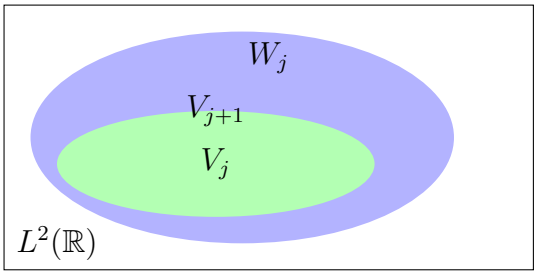
Ett problem återstår dock ännu: V_j innehåller alla de funktioner som hör till V_{j-1}. Det här betyder att en approximation kan bestå till stora delar av enkla funktioner som hör till lägre approximationsrum än den aktuella skalan. För att kunna analysera vilka delar av en signal som behöver vilken noggrannhet behövs ett sätt att skilja approximationsrummen åt. Det är här krusningarna kommer in i bilden.

1.5 Krusningarna i Haarsystemet

Eftersom $V_j \subset V_{j+1}$ finns det funktioner $f \in V_{j+1}$ som är ortogonala till alla funktioner i V_j. Det går därför att dela upp V_{j+1} i två ortogonala delar: V_j och $V_j^\perp$, där $V_j^\perp$ betecknar det *ortogonala komplementet* till V_j och består av alla funktioner $f \in V_{j+1}$ sådana att $\langle f, g \rangle = 0$, $\forall g \in V_j$.

Definition 1.12. Låt W_j beteckna det ortogonala komplementet till V_j i V_{j+1}. Då kallas en samling funktioner som tillsammans utgör en bas för W_j för *krusningar*.

Krusningarna är alltså dyadiska stegfunktioner i V_{j+1} men ortogonala till alla funktioner i V_j. Det här betyder att varje $f \in W_j$ måste ha åtminstone ett dyadiskt intervall $I_{j,k}$ där f ”krusar sig”, d.v.s. ett intervall där f inte



Figur 1.6: Den blåa och den gröna ellipsen bildar tillsammans V_{j+1} medan V_j består av endast den gröna. Det blåa området utgör skillnaden mellan V_{j+1} och V_j d.v.s. W_j.

är konstant. Om f var konstant på varje $I_{j,k}$ skulle det betyda att f låg i V_j vilket definitionen på W_j inte tillåter! Enligt den allmänna teorin för Hilbertrum är W_j ett Hilbertrum försett med inreprodukten ur $L^2(\mathbb{R})$.

För att visa att V_{j+1} kan byggas upp med hjälp av V_j och W_j kan relationen mellan de olika vektorrummen skrivas som $V_{j+1} = V_j \oplus W_j$, där $\oplus$ betecknar direkta summan. Genom att dela upp $L^2(\mathbb{R})$ på det här sättet fås en *multiresolutionsanalys* (*eng.* multi resolution analysis) eller mångupplösningsanalys som kan användas för att analysera och hantera signaler på flera olika resolutionsnivåer. Se kapitel 4.

För att kunna göra uppdelningen av V_{j+1} behövs först en bas för W_j och en för V_j. En bas för V_j ges direkt av sats 1.11. Då återstår det alltså att hitta eller konstruera en bas för W_j.

Exempel 1.13. Vi undersöker om $p_{j+1,k} - p_{j,h}$ ligger i W_j med $h = \lfloor \frac{k}{2} \rfloor$.

- $p_{j+1,k} - p_{j,h}$

$=$ $\left\{ \begin{array}{l} \text{Använd den alternativa definitionen på skalningsfunktionerna} \\ \text{(lemma 1.5). Enligt lemma 1.6 gäller } I_{j+1,k} \subset I_{j,h}. \end{array} \right\}$

$$\begin{cases} 2^{\frac{j+1}{2}} - 2^{\frac{j}{2}} & \text{, då } x \in I_{j+1,k} \\ -2^{\frac{j}{2}} & \text{, då } x \in I_{j,h} \setminus I_{j+1,k} \\ 0 & \text{, annars.} \end{cases}$$

$=$ {Bryt ut $2^{\frac{j}{2}}$ ur det översta uttrycket.}

$$\begin{cases} 2^{\frac{j}{2}} \cdot (\sqrt{2} - 1) & \text{, då } x \in I_{j+1,k} \\ -2^{\frac{j}{2}} & \text{, då } x \in I_{j,h} \setminus I_{j+1,k} \\ 0 & \text{, annars.} \end{cases}$$

Beroende på om k är udda eller jämnt är $I_{j,h} \setminus I_{j+1,k}$ antingen $I_{j+1,k+1}$ eller $I_{j+1,k-1}$ (se lemma 1.6 och fig. 1.5). Det här betyder att $p_{j+1,k} - p_{j,h}$ ligger i V_{j+1} men inte i V_j eftersom funktionen inte är konstant på $I_{j,h}$. Enligt definitionen måste då $p_{j+1,k} - p_{j,h}$ ligga i W_j.

Som nämnts i början på det här kapitlet så går det att konstruera en krusningsbas utgående från en fader- och en moderkrusning. Faderkrusningen är den funktion $p(x)$ som användes för att bygga upp skalningsfunktionerna och moderkrusningen är $H(x)$ i definition 1.1. På samma sätt som $p_{j,k}(x) = 2^{\frac{j}{2}} p(2^j x - k)$ ger en bas för V_j går det också att skapa $H_{j,k}(x)$ för att få en bas för W_j.

$$H_{j,k}(x) = 2^{\frac{j}{2}} H(2^j x - k) \tag{1.1}$$

Det finns också ett alternativt sätt att skriva $H_{j,k}(x)$ på motsvarande sätt som lemma 1.5 för $p_{j,k}(x)$.

Lemma 1.14. *Ett annat sätt att skriva $H_{j,k}(x)$ är*

$$H_{j,k}(x) = \begin{cases} 2^{\frac{j}{2}} & , \textit{då } x \in I_{j+1,2k} \\ -2^{\frac{j}{2}} & , \textit{då } x \in I_{j+1,2k+1} \\ 0 & , \textit{annars.} \end{cases}$$

Bevis.

- $H_{j,k}$

$=$ {Ekvation 1.1 ger definitionen på $H_{j,k}$.}

$2^{\frac{j}{2}} H(2^j x - k)$

$=$ {Definition 1.1 ger uttrycket för $H(x)$.}

$$\begin{cases} 2^{\frac{j}{2}} & , \text{då } 2^j x - k \in [0, \frac{1}{2}) \\ -2^{\frac{j}{2}} & , \text{då } 2^j x - k \in [\frac{1}{2}, 1) \\ 0 & , \text{annars} \end{cases}$$

$=$ { Lös $2^j x - k \in [0, \frac{1}{2})$ och $2^j x - k \in [\frac{1}{2}, 1)$ för x och skriv uttrycken i form av dyadiska intervall. }

- $0 \leq 2^j x - k < \frac{1}{2}$

$\Leftrightarrow$ {Lös ut x.}

$$\frac{k}{2^j} \leq x < \frac{k + \frac{1}{2}}{2^j}$$

$\Leftrightarrow$ {Förläng vänster- och högerled med 2.}

$$\frac{2k}{2^{j+1}} \leq x < \frac{2k+1}{2^{j+1}}$$

$\Leftrightarrow$ {Använd definitionen för dyadiskt intervall (def. 1.2).}

$$x \in I_{j+1,2k}$$

- $\frac{1}{2} \leq 2^j x - k < 1$

$\Leftrightarrow$ {Lös olikheten och förläng på samma sätt som ovan.}

$$\frac{2k+1}{2^{j+1}} \leq x < \frac{2k+2}{2^{j+1}}$$

$\Leftrightarrow$ {Använd definitionen för dyadiskt intervall (def. 1.2).}

$$x \in I_{j+1,2k+1}$$

$$\ldots \quad \begin{cases} 2^{\frac{j}{2}} & \text{, då } x \in I_{j+1,2k} \\ -2^{\frac{j}{2}} & \text{, då } x \in I_{j+1,2k+1} \\ 0 & \text{, annars} \end{cases}$$

□

Varje funktion $H_{j,k}(x)$ är alltså den dyadiska stegfunktion i V_{j+1} som fås genom att dela den positiva biten av $p_{j,k}$ på mitten och byta tecken på den högra delen.

Sats 1.15. *Funktionsserien* $\{H_{j,k}(x)\}_{k=-\infty}^{\infty}$ *är en ortonormerad bas i* W_j.

Bevis. Beviset är uppdelat i tre delar på samma sätt som i sats 1.11.

- Visa att $H_{j,k}$ är normerad.

$\Vdash$ {Serien $H_{j,k}$ är normerad omm $\langle H_{j,k}, H_{j,k} \rangle = 1$.}

$$\langle H_{j,k}, H_{j,k} \rangle$$

$=$ {Använd inreproduktens definition.}

$$\int_{-\infty}^{\infty} H_{j,k}(x) H_{j,k}(x) dx$$

$=$ { Dela upp integralen och förenkla på samma sätt som i beviset till sats 1.11. }

$$\int_{\frac{k}{2^j}}^{\frac{k+\frac{1}{2}}{2^j}} 2^{\frac{j}{2}} \cdot 2^{\frac{j}{2}} dx + \int_{\frac{k+\frac{1}{2}}{2^j}}^{\frac{k+1}{2^j}} (-2^{\frac{j}{2}}) \cdot (-2^{\frac{j}{2}}) dx$$

$=$ $\left\{ \begin{array}{l} \text{Utför multiplikationerna innuti integralerna och slå ihop in-} \\ \text{tegralerna.} \end{array} \right\}$

$$\int_{\frac{k}{2^j}}^{\frac{k+1}{2^j}} 2^j dx$$

$=$ {Utför integralen.}

$$2^j \left(\frac{k+1}{2^j} - \frac{k}{2^j} \right)$$

$=$ {Förenkla uttrycket.}

$$1$$

■

- Visa att $\{H_{j,k}\}_{k=-\infty}^{\infty}$ är ortogonal.

(a) $k < m$

$\Vdash$ {Följden $\{H_{j,k}\}_{k=-\infty}^{\infty}$ är ortogonal omm $\langle H_{j,k}, H_{j,m} \rangle = 0$ för $k \neq m$.}

$$\langle H_{j,k}, H_{j,m} \rangle$$

$=$ {Använd inreproduktens definition.}

$$\int_{-\infty}^{\infty} H_{j,k}(x) H_{j,m}(x) dx$$

$=$ $\left\{ \begin{array}{l} \text{Dela upp på samma sätt som ovan och sätt in värden på } H_{j,k} \\ \text{och } H_{j,m}. \text{ Enligt } (a) \text{ gäller } k < m. \end{array} \right\}$

$$\int_{\frac{k}{2^j}}^{\frac{k+\frac{1}{2}}{2^j}} 2^{\frac{j}{2}} \cdot 0 dx + \int_{\frac{k+\frac{1}{2}}{2^j}}^{\frac{k+1}{2^j}} (-2^{\frac{j}{2}}) \cdot 0 dx + \int_{\frac{k+1}{2^j}}^{\frac{m}{2^j}} 0 dx + \int_{\frac{m}{2^j}}^{\frac{m+\frac{1}{2}}{2^j}} 0 \cdot 2^{\frac{j}{2}} dx + \int_{\frac{m+\frac{1}{2}}{2^j}}^{\frac{m+1}{2^j}} 0 \cdot (-2^{\frac{j}{2}}) dx$$

$=$ {Alla termerna är 0.}

$$0$$

■

- Visa att $\{H_{j,k}\}_{k=-\infty}^{\infty}$ är en bas i W_j.

(a) $f \in W_j$

$\Vdash$ {Följden $\{H_{j,k}\}_{k=-\infty}^{\infty}$ är en bas i W_j omm $\langle H_{j,k}, f\rangle = 0 \Leftrightarrow f = 0$.}

$\langle H_{j,k}, f\rangle = 0$

$\Leftrightarrow$ {Använd inreproduktens definition.}

$$\int_{-\infty}^{\infty} H_{j,k}(x) f(x) dx = 0$$

$\Leftrightarrow$ {Dela upp, förenkla och sätt in värdet på $H_{j,k}(x)$ i integralen.}

$$\int_{\frac{k}{2^j}}^{\frac{k+\frac{1}{2}}{2^j}} 2^j f(x) dx + \int_{\frac{k+\frac{1}{2}}{2^j}}^{\frac{k+1}{2^j}} (-2^j) f(x) dx = 0$$

$\Leftrightarrow$ {Bryt ut 2^j.}

$$2^j \left(\int_{\frac{k}{2^j}}^{\frac{k+\frac{1}{2}}{2^j}} f(x) dx - \int_{\frac{k+\frac{1}{2}}{2^j}}^{\frac{k+1}{2^j}} f(x) dx \right) = 0$$

$\Leftrightarrow$ { Antagande (a) betyder att f måste vara konstant på varje dyadiskt intervall med skalan $j+1$. Genom att förlänga integrationsgränserna med 2 fås just sådana intervall. }

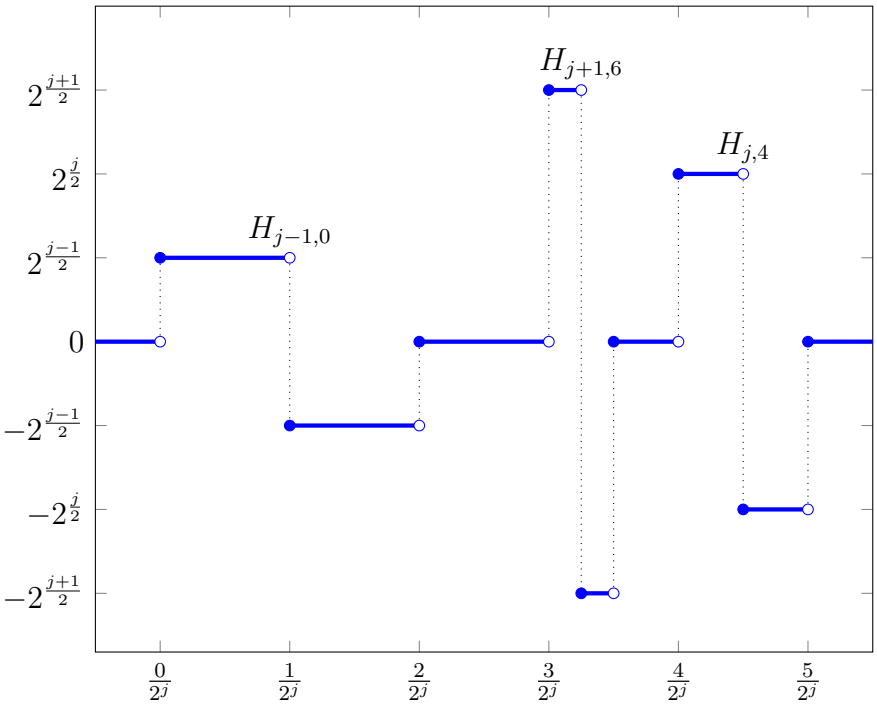
$$2^j \left(\int_{\frac{2k}{2^{j+1}}}^{\frac{2k+1}{2^{j+1}}} f(x) dx - \int_{\frac{2k+1}{2^{j+1}}}^{\frac{2k+2}{2^{j+1}}} f(x) dx \right) = 0$$

$\Leftrightarrow$ { Integrationsintervallen är nu $I_{j+1,2k}$ respektive $I_{j+1,2k+1}$, alltså två på varandra följande intervall. För att ekvationen ska gälla måste f ha samma värde på båda intervallen men det skulle betyda att f är konstant på $I_{j,k}$ och då ligger f i V_j vilket inte är tillåtet enligt (a)! Den enda återstående möjligheten är att $f = 0$. }

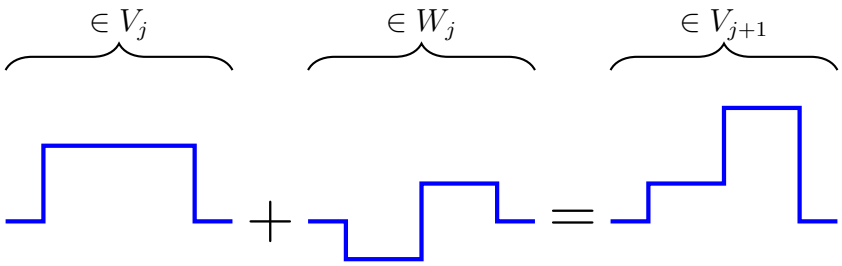
$f = 0$

■ □

Med hjälp av baserna i V_j och W_j går det nu att uttrycka en godtycklig funktion $f \in V_{j+1}$ som en linjär kombination av termer ur $\{p_{j,k}(x)\}_{k=-\infty}^{\infty}$ och $\{H_{j,k}(x)\}_{k=-\infty}^{\infty}$ (se fig. 1.8). En funktion som till största delen ligger i V_j behöver inte speciellt många termer ur $\{H_{j,k}(x)\}_{k=-\infty}^{\infty}$ vilket ger komprimeringsmöjligheter.



Figur 1.7: Tre krusningar med olika skalor och position. Jämför med skalningsfunktionerna i figur 1.4.



Figur 1.8: En funktion i V_{j+1} kan uttryckas med hjälp av funktioner i V_j och W_j.

Kapitel 2

Att dela upp en signal i beståndsdelar

Som tidigare nämnts har krusningar många användningsområden. Orsaken till att krusningarna är så användbara är att det med hjälp av dem går att dela upp en signal i V_j i komponenter enligt underrummen till V_j, ungefär på samma sätt som kartor i olika skala över ett och samma område.

För ett stort j kan signalen i V_j då jämföras med en detaljkarta medan ett mindre j ger motsvarigheten till en översiktskarta där bara de största strukturerna finns med.

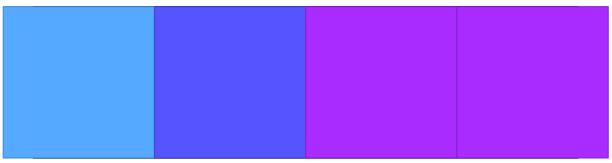
Till skillnad från kartorna lagras dock inte hela signalen för varje upplösningsnivå utan endast skillnaderna (d.v.s. den del av signalen i V_{j+1} som finns i W_j) mellan nivåerna ända ner till den lägsta upplösningsnivån. Kartornas motsvarighet till det här är så kallade kartlager som kan läggas till en grundkarta. Ett lager kan t.ex. innehålla information om var det finns busshållplatser. Skilt för sig är ett kartlager knappast speciellt användbart, men tillsammans med grundkartan och andra kartlager går det att sätta ihop en skräddarsydd karta för något specifikt behov. Uppdelningen i olika lager gör det möjligt att få fram olika kombinationer av information utan att behöva lagra varje karta skilt för sig.

Förutom uppdelningen av signaler presenteras här också tolkningen av signalerna som bilder för att förbereda tillämpningen av krusningar för bildkomprimering i följande kapitel.

2.1 Bilder i 1D

En digital bild består av ett bestämt antal punkter (pixlar) med numeriska värden som anger färgen i varje given punkt. Det finns flera olika sätt att

representera färger med hjälp av tal, så datorn måste veta vilket system som används för att färgerna ska bli rätt. Ett normalt digitalt foto kan bestå av flera miljoner pixlar, vilket gör det svårt att hantera informationen och bearbeta den för hand. För att undersöka krusningarna och de möjligheter de ger upphov till behövs någonting enklare att utgå från. Låt därför vektorn $B = \begin{pmatrix} 2 & 4 & 5 & 5 \end{pmatrix}$ representera en enkel endimensionell bild (se fig. 2.1).



(a) En färgskala där 2 motsvarar ljusblått och 5 lila.

(b) Värdena tolkas här med en skala där 0 är svart och 6 är vitt.

Figur 2.1: Vektorn B ritad i nyanser av blått respektive grått.

Genom att tolka B som en dyadisk stegfunktion $B(x)$, vars stöd utgörs av intervallet $[0, 1]$, fås följande uttryck:

$$B(x) = \begin{cases} 2 & , 0 \leq x < \frac{1}{4} \\ 4 & , \frac{1}{4} \leq x < \frac{1}{2} \\ 5 & , \frac{1}{2} \leq x < \frac{3}{4} \\ 5 & , \frac{3}{4} \leq x < 1. \end{cases}$$

Med den här tolkningen är B alltså en funktion i V_2 och går då att skriva med hjälp av Haarkrusningens skalningsfunktioner (def. 1.3) i V_2 som utgör en bas för rummet.

$$B(x) = 2 \cdot \frac{p_{2,0}(x)}{2^2} + 4 \cdot \frac{p_{2,1}(x)}{2^2} + 5 \cdot \frac{p_{2,2}(x)}{2^2} + 5 \cdot \frac{p_{2,3}(x)}{2^2} \tag{2.1}$$

Eftersom $V_2 \subset V_3 \subset \ldots$ går det också att använda skalningsfunktioner från högre rum, men det skulle ge upphov till flera termer och är således onödigt och kontraproduktivt vad gäller komprimering.

Anmärkning 2.1. Genom att välja intervallet $[0, 4)$ istället för $[0, 1)$ kunde konstanterna $\frac{1}{2^2}$ undvikas för tillfället, men då signalen senare delas upp med hjälp av Haarkrusningarna behövs det ändå liknande konstanter.

För att komprimera signalen (bilden) gäller det först att dela upp den i komponenter bestående av krusningar. Uttrycket för $B(x)$ i ekvation 2.1 ger en uppdelning i skalningsfunktioner som kan utnyttjas som i figur 1.8.

2.2 Krusningstransformen för Haarsystemet

Definition 2.2. *Krusningstransformen* av en signal $s \in V_j$ är signalen uttryckt som en skalningsfunktion och en serie krusningar, så att skalningsfunktionens stöd är det samma som stödet för s,. För att visa på vilken krusningsbas som används kan krusningstransformen för Haarsystemet kallas Haartransformen.

Exempel 2.3. Vi skriver de två första termerna i $B(x)$ med hjälp av en krusning i Haarsystemet och en skalningsfunktion i V_1. Uppgiften är alltså att bestämma funktionerna $p_{1,k}$ och $H_{1,m}$, samt konstanterna k_1 och k_2 så att följande ekvation gäller.

$$2 \cdot \frac{p_{2,0}(x)}{2^2} + 4 \cdot \frac{p_{2,1}(x)}{2^2} = k_1 \cdot p_{1,k}(x) + k_2 \cdot H_{1,m}(x) \tag{2.2}$$

Först gäller det att hitta den skalningsfunktion $p_{1,k}$ som är positiv i samma intervall som $p_{2,0}$ och $p_{2,1}$. Det här kan vi göra med hjälp av lemma 1.6 som säger att $I_{2,0} \subset I_{1,0}$ och $I_{2,1} \subset I_{1,0}$ eftersom $0 = \lfloor \frac{0}{2} \rfloor = \lfloor \frac{1}{2} \rfloor$. Skalningsfunktionen som vi söker ska alltså vara positiv på $I_{1,0}$ vilket betyder att det måste vara fråga om $p_{1,0}$.

Lemma 1.14 ger att krusningen på intervallet $I_{j,k}$ är $H_{j,k}$ vilket betyder att krusningen vi söker är $H_{1,0}$. De två konstanterna som återstår att bestämma kan beräknas ur ekvationssystemet som fås då man delar upp ekvation 2.2 i intervallen $I_{2,0}$ och $I_{2,1}$.

- $2 \cdot \frac{p_{2,0}(x)}{2^2} + 4 \cdot \frac{p_{2,1}(x)}{2^2} = k_1 \cdot p_{1,0}(x) + k_2 \cdot H_{1,0}(x)$

$\Leftrightarrow$ {Dela upp ekvationen i intervallen $I_{2,0}$ och $I_{2,1}$.}

$$\begin{cases} 2 \cdot \frac{p_{2,0}(x)}{2^2} + 4 \cdot \frac{p_{2,1}(x)}{2^2} = k_1 \cdot p_{1,0}(x) + k_2 \cdot H_{1,0}(x) & , x \in I_{2,0} \\ 2 \cdot \frac{p_{2,0}(x)}{2^2} + 4 \cdot \frac{p_{2,1}(x)}{2^2} = k_1 \cdot p_{1,0}(x) + k_2 \cdot H_{1,0}(x) & , x \in I_{2,1} \end{cases}$$

$\Rightarrow$ {Evaluera funktionerna och förenkla.}

$$\begin{cases} 2 \cdot 1 + 4 \cdot 0 & = k_1 \cdot 2 + k_2 \cdot 2^{\frac{1}{2}} \\ 2 \cdot 0 + 4 \cdot 1 & = k_1 \cdot 2 + k_2 \cdot (-2^{\frac{1}{2}}) \end{cases}$$

$\Leftrightarrow$ {Lös ut k_1 och k_2.}

$$\begin{cases} k_1 &= \frac{3}{2} \\ k_2 &= -\frac{1}{\sqrt{2}} \end{cases}$$

Slutresultatet blir alltså $2 \cdot \frac{p_{2,0}(x)}{2^2} + 4 \cdot \frac{p_{2,1}(x)}{2^2} = \frac{3}{2} \cdot p_{1,0}(x) - \frac{1}{\sqrt{2}} \cdot H_{1,0}(x)$, eller med konstanterna i det högra ledet skrivna på samma sätt som i det vänstra:

$$2 \cdot \frac{p_{2,0}(x)}{2^2} + 4 \cdot \frac{p_{2,1}(x)}{2^2} = 3 \cdot \frac{p_{1,0}(x)}{2} - 1 \cdot \frac{H_{1,0}(x)}{\sqrt{2}}.$$

Ingen information går förlorad i proceduren och ingenting nytt tillkommer heller. Den transformerade vektorn $B_t = \begin{pmatrix} 3 & -1 & 5 & 0 \end{pmatrix}$, som fås genom att utföra proceduren i exemplet för samtliga termer, påminner inte direkt om den ursprungliga men det finns ändå en klar koppling mellan dem: konstanten k_1 i exemplet är medeltalet av 2 och 4 och k_2 är skillnaden mellan 2 och k_1. På samma sätt gäller det för 5 och 0 att $5 = \frac{5+5}{2}$ och $0 = 5 - 5$. Det här är ingen tillfällighet utan faktiskt en regel som kan användas för att dela upp en signal i V_{j+1} i komponenter ur V_j och W_j.

Lemma 2.4. *Låt $s(x)$ vara en (del av en) signal i V_{j+1} uttryckt i basen av Haarsystemets skalningsfunktioner på följande vis:*

$$s(x) = k_1 \cdot \frac{p_{j+1,2k}(x)}{2^{j+1}} + k_2 \cdot \frac{p_{j+1,2k+1}(x)}{2^{j+1}}.$$

Haartransformen för signalen ges då av

$$\frac{k_1 + k_2}{2} \cdot \frac{p_{j,k}(x)}{2^j} + \frac{k_1 - k_2}{2} \cdot \frac{H_{j,k}(x)}{2^{\frac{j}{2}}}.$$

Bevis.

- Visa att

$$k_1 \cdot \frac{p_{j+1,2k}(x)}{2^{j+1}} + k_2 \cdot \frac{p_{j+1,2k+1}(x)}{2^{j+1}} = \frac{k_1 + k_2}{2} \cdot \frac{p_{j,k}(x)}{2^j} + \frac{k_1 - k_2}{2} \cdot \frac{H_{j,k}(x)}{2^{\frac{j}{2}}}.$$

$\Vdash$ $k_1 \cdot \frac{p_{j+1,2k}(x)}{2^{j+1}} + k_2 \cdot \frac{p_{j+1,2k+1}(x)}{2^{j+1}}$

$=$ {Dela upp signalen och förenkla.}

$$\begin{cases} k_1 & , x \in I_{j+1,2k} \\ k_2 & , x \in I_{j+1,2k+1} \\ 0 & , annars \end{cases}$$

$$= \quad \left\{ \begin{array}{l} \text{Skalningsfunktionen } p_{j,k} \text{ har värdet } 2^j \text{ på intervallet } I_{j,k} \text{ d.v.s.} \\ \text{på både } I_{j+1,2k} \text{ och } I_{j+1,2k+1}. \text{ Skriv } k_i \text{ som } \frac{k_i+k_j}{2} + \frac{k_i-k_j}{2}. \end{array} \right\}$$

$$\begin{cases} \frac{k_1+k_2}{2} \cdot \frac{p_{j,k}(x)}{2^j} + \frac{k_1-k_2}{2} & , x \in I_{j+1,2k} \\ \frac{k_1+k_2}{2} \cdot \frac{p_{j,k}(x)}{2^j} - \frac{k_1-k_2}{2} & , x \in I_{j+1,2k+1} \\ 0 & , annars \end{cases}$$

$$= \quad \{\text{Krusningen } H_{j,k} \text{ tar värdet } 2^{\frac{j}{2}} \text{ på } I_{j+1,2k} \text{ och } -2^{\frac{j}{2}} \text{ på } I_{j+1,2k+1}.\}$$

$$\begin{cases} \frac{k_1+k_2}{2} \cdot \frac{p_{j,k}(x)}{2^j} + \frac{k_1-k_2}{2} \cdot \frac{H_{j,k}(x)}{2^{\frac{j}{2}}} & , x \in I_{j+1,2k} \\ \frac{k_1+k_2}{2} \cdot \frac{p_{j,k}(x)}{2^j} + \frac{k_1-k_2}{2} \cdot \frac{H_{j,k}(x)}{2^{\frac{j}{2}}} & , x \in I_{j+1,2k+1} \\ 0 & , annars \end{cases}$$

$$= \quad \left\{ \begin{array}{l} \text{Både } p_{j,k} \text{ och } H_{j,k} \text{ är 0 utanför } I_{j,k}. \text{ Det går med andra ord} \\ \text{att slå ihop de tre olika fallen till ett enda uttryck.} \end{array} \right\}$$

$$\frac{k_1 + k_2}{2} \cdot \frac{p_{j,k}(x)}{2^j} + \frac{k_1 - k_2}{2} \cdot \frac{H_{j,k}(x)}{2^{\frac{j}{2}}}$$

■ □

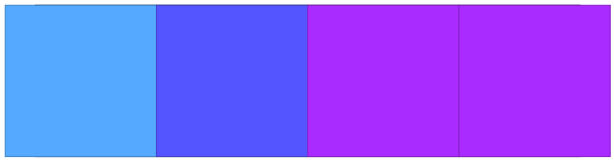
Komponenterna som fås består av en approximation av signalen i V_j, med lägre upplösning än den ursprungliga, och en del med ”detaljerna” i W_j.

Approximationen består av den serie som fås genom att ersätta varje par av termer $k_{2m} \cdot p_{j+1,2m}(x)$ och $k_{2m+1} \cdot p_{j+1,2m+1}(x)$ med en ny som utgör medeltalet av de båda termerna över $I_{j,m}$. I vektorform motsvarar det här helt enkelt att minska vektorn till halva storleken genom att ersätta varje talpar med medeltalet: $\begin{pmatrix} 2 & 4 & 5 & 5 \end{pmatrix} \rightarrow \begin{pmatrix} 3 & 5 \end{pmatrix}$. För bilder innebär det här det samma som att minska upplösningen med hälften.

Om signalen i vektorform inte har 2^n kolonner för något $n \in \mathbb{N}$ uppstår det problem förr eller senare då det inte går att halvera jämnt. Det går ändå att kringgå problematiken genom att lägga till nollor i endera ändan.

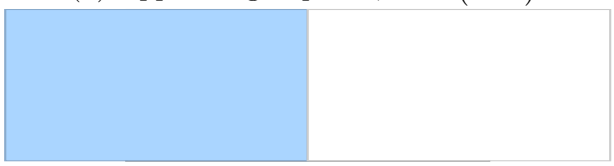
Den andra delen av signalen består av skillnaderna mellan approximationen och den ursprungliga signalen och kan tolkas som det fel som uppstår om den ursprungliga signalen ersätts med approximationen.

För att fullfölja krusningstransformen måste den grova signalen delas upp ytterligare på samma sätt ända tills alla termer utom en beskrivs av krusningar. Då har bilden reducerats till en enda pixel innehållande medeltalet av alla de ursprungliga pixlarna.



(a) Upplösning: 4 pixlar, $B = \begin{pmatrix} 2 & 4 & 5 & 5 \end{pmatrix}$.

(b) Upplösning: 2 pixlar, $B = \begin{pmatrix} 3 & 5 \end{pmatrix}$.



(c) Detaljvektorn $\begin{pmatrix} -1 & 0 \end{pmatrix}$.

(d) Krusningstransformen av B: $\begin{pmatrix} 4 & -1 & -1 & 0 \end{pmatrix}$.

Figur 2.2: Bilden B ritad med olika upplösning, detaljvektorn och den fullständiga krusningstransformen. Vektorerna med två pixlar är skalade till samma bredd som den med fyra pixlar för att det ska gå lättare att jämföra dem. Detaljvektorn och krusningstransformen är ritade för absolutbeloppen på sina värden för att följa färgskalans gradering.

Medeltal	Detaljkoefficienter
$\begin{pmatrix}2 & 4 & 5 & 5\end{pmatrix}$	
$\begin{pmatrix}3 & 5\end{pmatrix}$	$\begin{pmatrix}-1 & 0\end{pmatrix}$
$\begin{pmatrix}4\end{pmatrix}$	$\begin{pmatrix}-1\end{pmatrix}$

Haartransformen (i vektorform) av B är vektorn som fås genom att börja från den grövsta signalen och sedan lägga till detaljkoefficienterna med början på den lägsta nivån.

$$\begin{pmatrix}2 & 4 & 5 & 5\end{pmatrix} \to \begin{pmatrix}4 & -1 & -1 & 0\end{pmatrix}$$

Krusningstransformen av en vektor är oanvändbar utan information om vilket krusningssystem som använts för att skapa den eftersom systemet säger hur koefficienterna ska tolkas. För enkelhetens skull är det ändå ofta bäst att hantera signalerna i vektorform. Med alla funktioner utskrivna ser Haartransformen av B ut så här:

$$4 \cdot p_{0,0}(x) - 1 \cdot H_{0,0}(x) - 1 \cdot \frac{H_{1,0}(x)}{\sqrt{2}} + 0 \cdot \frac{H_{1,1}(x)}{\sqrt{2}}.$$

2.2.1 Beräkning av Haartransformen i Octave

Eftersom bilder i allmänhet består av ett stort antal pixlar blir det snabbt opraktiskt att beräkna krusningstransformerna för hand. Det kan då löna sig att utnyttja t.ex. Octave för att automatisera arbetet.

Funktionen `wave(x,num)` beräknar transformen av en vektor `x` ett steg i taget fram till och med steg `num`. För att snabba upp beräkningarna och spara minne används en så kallad *in situ*-algoritm (lat., 'på stället') vilket innebär att koefficienterna som beräknas ersätter de ursprungliga värdena istället för att sparas i en separat vektor. Implementationen finns i exempelkod A.1.

Exempel 2.5. Vi beräknar Haartransformen av vektorn $A = \begin{pmatrix}a & b\end{pmatrix}$ in situ.

Då man räknar in situ måste man använda de värden som för tillfället finns lagrade i vektorn, inte de ursprungliga. Låt därför A_1 och A_2 beteckna värdet på plats 1 respektive 2 i A.

- $A = \begin{pmatrix}a & b\end{pmatrix}$

$\to$ $\{\text{Sätt } A_2 = \frac{A_1 - A_2}{2}\}$

$A = \begin{pmatrix}a & \frac{a-b}{2}\end{pmatrix}$

$\to$ $\{\text{Sätt } A_1 = A_1 - A_2\}$

$A = \begin{pmatrix}\frac{a+b}{2} & \frac{a-b}{2}\end{pmatrix}$

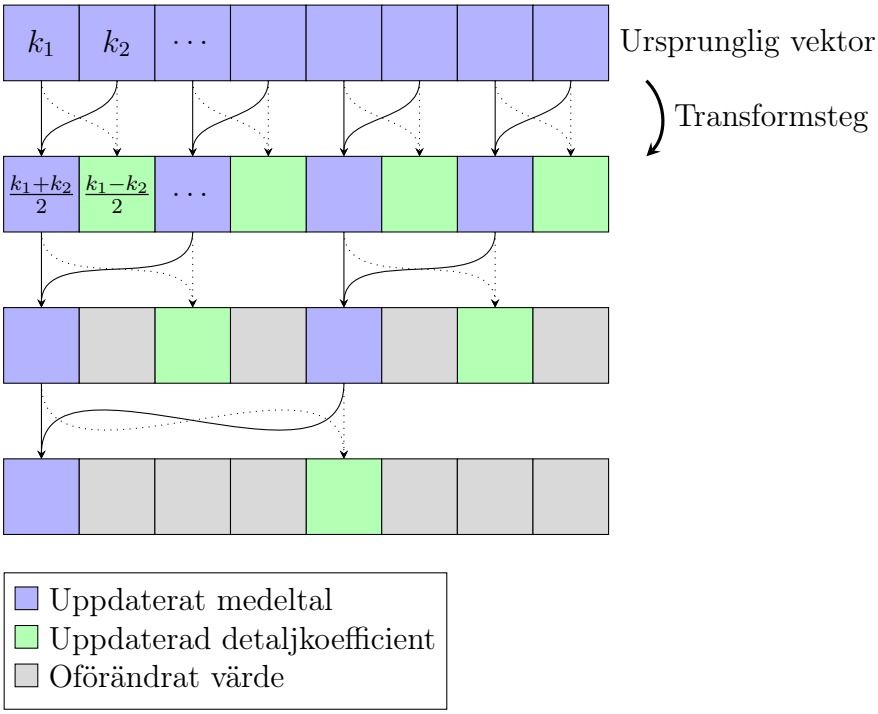
Att räkna på det här viset ger inga direkta fördelar vad gäller så här små vektorer, men för att transformera större mängder data kan det göra stor skillnad. Om beräkningen inte utförs in situ måste datorn göra en kopia av den ursprungliga vektorn för att kunna använda de värdena i beräkningarna. Eftersom kopian tar upp lika mycket minne som den ursprungliga vektorn kan det alltså krävas dubbelt så mycket minne för att köra ett sådant program jämfört med ett som gör beräkningarna in situ.

Exempel 2.6. Med samma vektor B som tidigare kan `wave` användas så här:

```
>> B = [2 4 5 5]
B =

   2   4   5   5

>> wave(B,1)
ans =

   3  -1   5   0

>> wave(B,2)
ans =

   4  -1  -1   0
```

På grund av in situ-algoritmen blir placeringen av värdena i den resulterande vektorn lite annorlunda än normalt (se fig. 2.3). Men det går ändå att få ut detaljkoefficienterna och medeltalen separat om det behövs.

```
>> [retur, medel, detalj] = wave(B,1)
retur =

   3  -1   5   0

medel =

   3   5

detalj =

  -1   0
```



Figur 2.3: Illustration över placeringen av värdena vid transformering in situ.

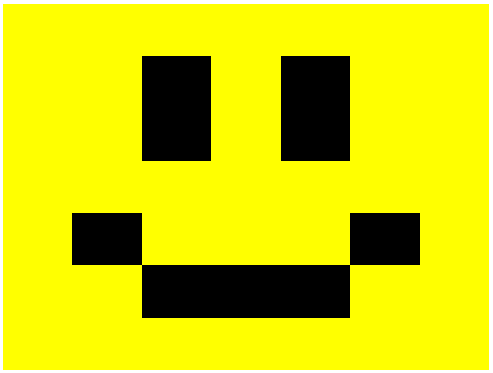
2.3 Haartransformen i 2D

Genom att använda matriser istället för vektorer fås vanliga tvådimensionella bilder. En riktigt enkel bild kunde t.ex. se ut som matrisen A här under.

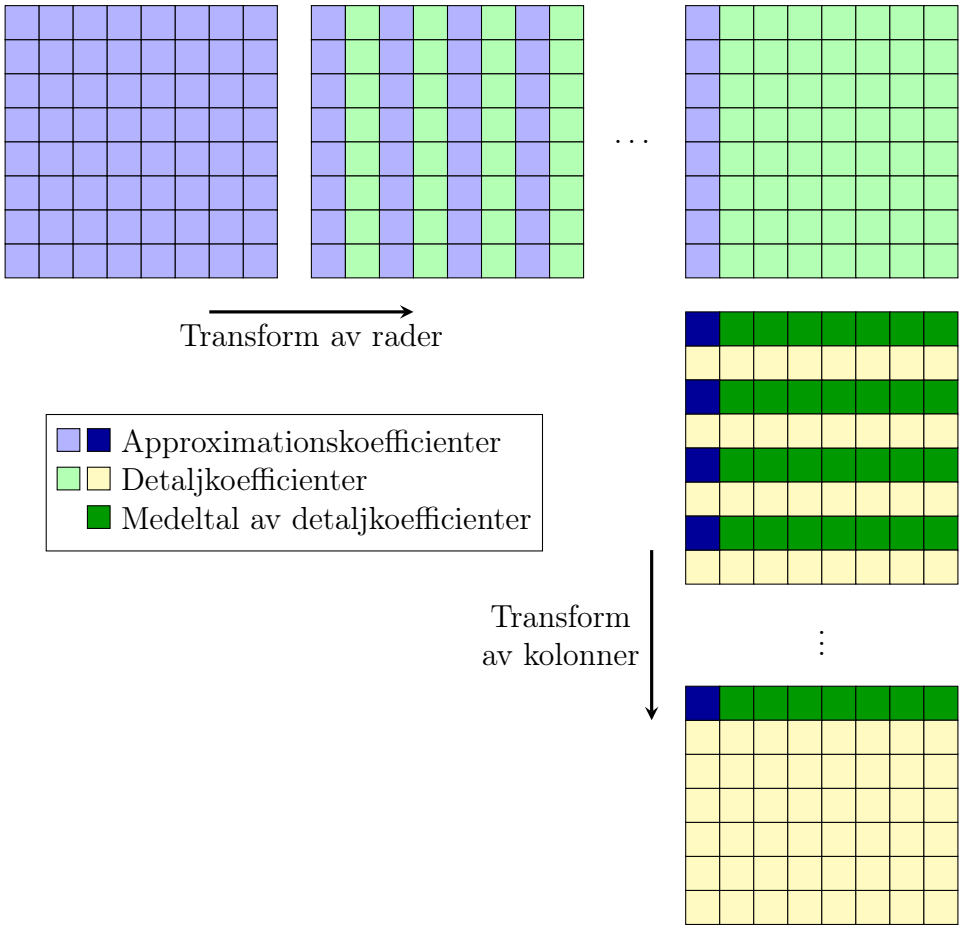
$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Genom att färga pixlarna med värde 0 gula och de med värde 1 svarta fås bilden i figur 2.4.

Haartransformen av en matris kan räknas på flera sätt men ett vanligt tillvägagångssätt är att först transformera varje rad i matrisen och sedan göra om samma sak för kolonnerna. En annan möjlighet är att först bara utföra ett steg i transformen för raderna och sedan ett steg för kolonnerna och fortsätta växla mellan dem på det sättet tills transformen är klar. Här kommer det första sättet att användas i form av funktionen `decompose` (exempelkod A.3).



Figur 2.4: En enkel tvådimensionell bild.



Figur 2.5: Transformering in situ i två dimensioner.

Efter en fullständig transform består bilden av en matris där elementet på plats $(1, 1)$ är ett medeltal för värdet på alla pixlar i hela bilden, resten av elementen i den översta raden är medeltal av detaljkoefficienterna i varje kolonn och alla andra element är detaljkoefficienter. För att få transformen att gå jämnt ut går det att lägga till extra rader eller kolonner av nollor vid behov.

Exempel 2.7. Bilden i figur 2.6 är 512 pixlar hög och 768 pixlar bred. Varje pixel har ett värde i intervallet $[0, 1]$ för att representera ljusintensiteten på så sätt att 0 är svart och 1 är vitt.

Figur 2.6: Två papegojor. (Fotograf: Steve Kelly. Källa: `http://r0k.us/graphics/kodak/kodim23.html`)

Det går att läsa in bilder i Octave och spara dem som matriser med kommandot `imread`. Med hjälp av `im2double` försäkrar vi oss om att matrisen består av flyttal mellan 0 och 1. För att visa matrisen som en bild kan funktionen `imshow` användas.

```
>> img = imread("standard pictures/papegojor.tiff");
>> img = im2double(img);
>> imshow(img);
```

För att kunna utföra Haartransformen på bilden måste både bredd och höjd vara potenser av 2. Det här kravet är redan uppfyllt för höjden, men bredden måste justeras till 1024 genom att lägga till kolonner med nollor (fig. 2.7).

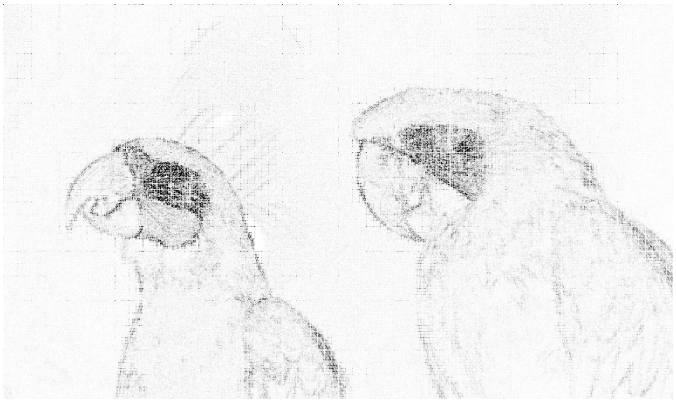
```
>> ex = ceil(log(columns(img))/log(2));
>> paddingSide = zeros(rows(img), pow2(ex) - columns(img));
>> img = [img, paddingSide];
>> imshow(img);
```

Bilden är nu klar för att transformeras med `decompose`.

Figur 2.7: Bilden är nu tillräckligt bred för transformen.

```
>> [decomposed, appr, detail] = decompose(img);
>> decomposed = decomposed(:,1:end-columns(paddingSide));
>> imshow(1-abs(decomposed).*15);
```

Den transformerade bilden består till största delen av pixlar med små värden vilket ger en nästan helt svart bild. Genom att justera färgskalan eller multiplicera matrisen med en konstant går det ändå att få en uppfattning om detaljkoefficienterna (se figur 2.8).



Figur 2.8: Bilden efter transformen med justerad intensitet och de extra kolonnerna borttagna. Ju mörkare punkt desto större är absolutbeloppet på koefficienten.

Tack vare in situ-algoritmen befinner sig en stor del av detaljkoefficienterna nära ”rätt ställe” vilket gör att den ursprungliga bilden går att känna igen. Jämför med figur 2.3! Hälften av detaljkoefficienterna från det första

steget finns kvar på sina ursprungliga platser ännu i det sista steget.

Till skillnad från elementen i den ursprungliga bildens matris så representerar värdena i den transformerade matrisen inte ljusintensitet utan istället skillnader i intensiteten mellan pixlarna i den ursprungliga bilden. Koefficienterna visar alltså var det förekommer stora variationer. Det här syns tydligt t.ex. kring fåglarnas ögon där det på den ursprungliga bilden fanns svart-vita ränder.

2.4 Transformens invers

För att det ska vara någon idé med krusningstransformen måste det också finnas ett sätt att komma tillbaka till den ursprungliga formen efter eventuella analyser eller modifikationer. Det här går att göra stegvis på samma sätt som i lemma 2.4 genom att lösa ut de ursprungliga koefficienterna ur transformuttrycket.

Enligt lemmat är koefficienterna efter transformen $t_1 = \frac{k_1+k_2}{2}$ och $t_2 = \frac{k_1-k_2}{2}$, där k_1 och k_2 är de ursprungliga koefficienterna. Genom att addera respektive subtrahera t_1 och t_2 kan de ursprungliga koefficienterna lösas ut.

$$t_1 + t_2 = \frac{k_1 + k_2}{2} + \frac{k_1 - k_2}{2} = k_1$$
$$t_1 - t_2 = \frac{k_1 + k_2}{2} - \frac{k_1 - k_2}{2} = k_2$$

Den inversa transformen kan också utföras in situ genom att följa proceduren i exempel 2.5 baklänges och invertera operatorerna.

- $A = \begin{pmatrix} \frac{a+b}{2} & \frac{a-b}{2} \end{pmatrix}$

$\rightarrow$ {Sätt $A_1 = A_1 + A_2$}

$A = \begin{pmatrix} a & \frac{a-b}{2} \end{pmatrix}$

$\rightarrow$ {Sätt $A_2 = A_1 - 2 \cdot A_2$}

$A = \begin{pmatrix} a & b \end{pmatrix}$

Funktionen `unwave(x,num)` (exempelkod A.2) är en implementation av den inversa transformen in situ och kan alltså användas för att få tillbaka den ursprungliga vektorn ur den transformerade.

```
>> A = [1 3 10 6];
>> wave(A,2)
ans =

   5  -1  -3   2

>> unwave(ans,2)
ans =

    1    3   10    6
```

På samma sätt som `unwave` fungerar som invers för `wave` har `decompose` sin invers i form av funktionen `recompose` (exempelkod A.4). För att återställa bilden i exemplet till sitt ursprungliga skick är det bara att köra `recompose` med den transformerade matrisen som input (igen med extra nollor för att transformen ska gå jämnt ut).

Kapitel 3

Bildkomprimering

I det här kapitlet kommer Haarsystemet att användas för bildkomprimering. Men principen går att tillämpa även på andra typer av data i matris- eller vektorform.

3.1 Principen

Komprimering med hjälp av krusningar bygger på att det i någon mån finns en koppling mellan värdena i en signal. Då det gäller bilder finns det ofta flera pixlar med ungefär samma färg nära varandra eftersom en stor del av föremålen omkring oss har relativt enhetliga färger. Det här innebär att bilden i matrisform har flera (nästan) likadana värden samlade nära varandra, vilket betyder att detaljkoefficienterna i Haartransformen blir 0 eller åtminstone små absolut sett. I exemplet med vektorn $B = \begin{pmatrix} 2 & 4 & 5 & 5 \end{pmatrix}$ blev den sista detaljkoefficienten 0 just på grund av att de två sista värdena i B är identiska.

En nolla bland detaljkoefficienterna tillför ingenting till signalen och små absoluta värden tillför inte så mycket. Därför går det att stryka en del av koefficienterna utan att signalen (bilden) förvrängs allt för mycket. Men det gäller ändå att hålla reda på koefficienternas positioner i vektorn eller matrisen för att kunna tolka den transformerade signalen rätt. Det går alltså inte att bara lämna bort några koefficienter. Med tillräckligt många nollor i en matris går det däremot att övergå till att lagra endast de element som är olika noll i en lista med positioner för varje värde, istället för att spara hela matrisen, och på så sätt åstadkomma komprimering.

Anmärkning 3.1. Att stryka eller ta bort koefficienter innebär här helt enkelt att sätta deras värde till 0.

För att komprimeringen ska lyckas gäller det alltså att få så många nollor

som möjligt i den transformerade signalen. I Haarsystemet uppstår nollorna då två närliggande värden är lika eller med andra ord då en del av en signal i V_j ligger helt i V_{j-1}. Skalningsfunktionerna fungerar på sätt och vis som ”gissningar” för hur signalen ser ut och om en gissning är bra behövs det inget tillägg för att representera signalen korrekt i följande approximationsrum. Olika krusningssystem har olika skalningsfunktioner och gör således olika gissningar vilket ger möjlighet till anpassning av komprimeringen till olika situationer.

3.2 Praktiken

För att komprimera en bild med hjälp av krusningar kan följande procedur användas:

1. Utför krusningstransformen på bilden.
2. Stryk önskvärt antal detaljkoefficienter.
3. Lagra bilden i den här formen.
4. Rekonstruera bilden med den inversa transformen.

Det andra steget är egentligen inte helt nödvändigt eftersom ett väl valt krusningssystem ser till att en stor del av detaljkoefficienterna blir 0. Komprimering utan att stryka några koefficienter kallas för icke-förstörande eftersom ingen information går förlorad. Genom att stryka koefficienter fås istället förstörande komprimering vilket är den typ som kommer att behandlas här.

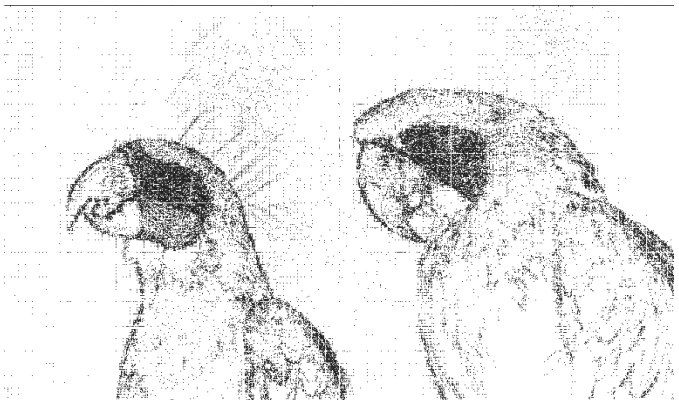
För att göra så liten skada som möjlig på bilden gäller det att stryka koefficienterna i växande ordning med början från den med minst absolutbelopp. Funktionen `compress(img, fact)` (kodexempel A.5) kan användas för att åstadkomma det här i Octave. Argumentet `fact` anger då andelen detaljkoefficienter som ska strykas som ett tal mellan 0 och 1, t.ex. skulle 0,7 innebära att 70% av de koefficienter som är olika 0 stryks.

Exempel 3.2. Som fortsättning på exempel 2.7 i föregående kapitel ska vi här komprimera bilden med de två papegojorna genom att ta bort 90% av detaljkoefficienterna ur den transformerade matrisen (d.v.s. 90% av de koefficienter som inte redan är 0 efter transformen). Efter komprimeringen med hjälp av `compress` går det att använda `spy` för att få en graf över de koefficienter som finns kvar i den glesa matrisen (se figur 3.1).

```
>> num = log(rows(decomposed))/log(2);
>> decomposed(1,:) = 0;
>> compressed = compress(decomposed, 0.9);
```

```
Compression factor: 90%
Elapsed time is 26.8026 seconds.
>> compressed(1,:) = appr;
>> spy(compressed(:,1:end-columns(paddingSide)));
```

För att se till att endast detaljkoefficienter stryks under komprimeringen sätts approximationskoefficienterna till 0. Efteråt går det att återställa dem från `appr` som erhölls i exempel 2.7 då bilden transformerades.



Figur 3.1: Den här bilden må se ut att vara nästan identisk med figur 2.8 men skenet bedrar! Varje pixel med ett värde olika 0 är här färgad svart medan pixlarna i den andra bilden har olika grå nyanser beroende på värde. Skulle figur 2.8 ha gjorts på samma sätt som den här så skulle nästan hela bilden ha varit svart.

Den komprimerade bilden bör lagras i sin transformerade form för att det ska vara någon idé med komprimeringen. Men för att vi ska ha någon glädje av att titta på den måste den transformeras tillbaka till normal form.

```
>> recomposed = recompose(compressed);
>> recomposed = recomposed(:,1:end-columns(paddingSide));
>> imshow(recomposed);
```

Resultatet syns i figur 3.2.

3.2.1 Färgbilder

En bild i gråskala går att lagra som en matris så att varje element i matrisen anger ljusintensiteten i just den punkten på en skala från 0 till 1. För färgbilder är det inte riktigt lika lätt eftersom det finns så många olika färgnyanser.

Figur 3.2: Den komprimerade bilden rekonstruerad.

Ett av de vanligaste sätten att representera färgbilder är att lagra tre värden per pixel istället för ett. Det första värdet anger då den röda intensiteten för pixeln medan det andra och tredje värdet anger den gröna respektive blåa intensiteten. Då de här grundfärgerna sedan kombineras fås flera miljoner olika färgnyanser.

I matrisform kan bilden uttryckas som en tredimensionell matris eller som tre vanliga matriser. För att komprimera en sådan här färgbild är det lättast att dela upp bilden i tre matriser, en för varje grundfärg, och sedan komprimera varje matris skilt för sig som vanligt.

Exempel 3.3. Bilden på de två papegojorna är ursprungligen en färgbild och kan sålunda delas upp i grundfärgerna röd, grön och blå som i figur 3.3.

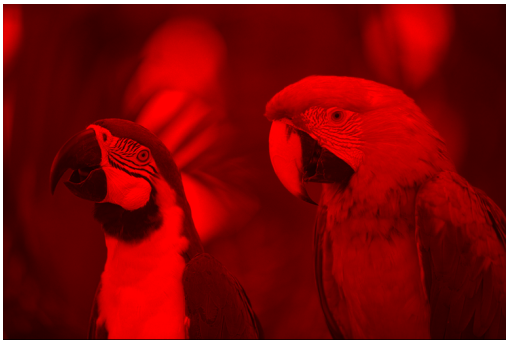
Genom att läsa in en färgbild med `imread` i Octave fås en tredimensionell matris som enkelt kan delas upp i tre olika matriser.

```
>> img = imread("standard␣pictures/kodim23.png");
>> img = im2double(img);
>> imgr = img(:,:,1); imgg = img(:,:,2); imgb = img(:,:,3);
```

Matriserna `imgr`, `imgg` och `imgb` kan nu tolkas som vanliga bilder i gråskala och komprimeras på samma sätt som sådana. Men för att få resultatet i figur 3.3 måste matriserna göras tredimensionella så att Octave kan tolka dem som färgbilder.

```
>> imgr = zeros(size(img)); imgg = zeros(size(img)); imgb = zeros(size(img));
>> imgr(:,:,1) = img(:,:,1); imgg(:,:,2) = img(:,:,2); imgb(:,:,3) = img(:,:,3);
```

Det går nu att se samma bilder som i figuren genom att använda `imshow` på de tre matriserna.



(a) Röd intensitet.

(b) Grön intensitet.

(c) Blå intensitet.

(d) Den ursprungliga bilden.

Figur 3.3: Bilden på papegojorna uppdelad i grundfärgerna röd, grön och blå.

3.3 Resultatet

För att det ska vara någon idé med komprimeringen måste det gå att se vad den komprimerade bilden föreställer och helst ska den avvika så lite som möjligt från den ursprungliga. Beroende på situation kan det ställas olika krav på hur mycket och på vilket sätt den resulterande bilden får avvika från originalet och därför behövs det också sätt att mäta de här avvikelserna.

Enklast av allt är kanske att rekonstruera bilden och inspektera den med egna ögon för att avgöra om den är tillräckligt bra. Om bilden då uppvisar för stora brister är det bara att göra om komprimeringen så att flera detaljkoefficienter hålls oförändrade.

I figur 3.4 finns ett exempel på hur de rekonstruerade bilderna kan se ut efter olika grader av komprimering. För att skapa figurerna har funktionen `compressRGB` använts och den i sin tur använder `compressImage` (se exempelkod A.7 och A.6). Koden är i princip bara sammansatt av de kommandon som har använts i tidigare exempel.

Bilden på papegojorna lämpar sig ganska bra för komprimering med Haarkrusningar eftersom den till stora delar består av sammanhängande, enfärgade områden utan små detaljer. Långt ifrån alla bilder uppfyller dock sådana kriterier, vilket kan leda till mycket sämre bildkvalitet för den komprimerade bilden trots samma komprimeringsgrad. Jämför t.ex. figur 3.5 med 3.4 för att se hur olika resultaten kan vara.

Istället för manuell inspektion av de komprimerade bilderna kan olika matrisnormer användas för att analysera skillnaderna mellan den komprimerade och den ursprungliga bilden.

Definition 3.4. För en $m \times n$-matris A definieras p-normen för $0 \leq p \leq \infty$ som

$$\|A\|_p = \max_{x:|x|_p=1} |Ax|_p,$$

där $|x|_p$ är p-normen för en vektor och x är en radvektor i $\mathbb{R}^n$.

I Octave kan p-normen beräknas för $p = 1, p = 2$ och $p = \infty$ med hjälp av den inbyggda funktionen `norm(A,p)`.

Exempel 3.5. Vi beräknar 2-normen för skillnaderna mellan de komprimerade och de ursprungliga bilderna i figur 3.4 och 3.5 i förhoppning om att normerna ska visa sig relatera till hur vi uppfattar bildkvaliteten.

Eftersom färgbilderna lagras som tre olika matriser måste vi räkna ut tre normer för varje komprimeringsgrad. För bilden på papegojorna ser beräkningarna ut på följande vis. Med 75% av detaljkoefficienterna strukna:

```
>> img = imread("standard pictures/kodim23.png");
>> img = im2double(img);
>> compressed = compressRGB(img,0.75);
>> diff = img - compressed;
>> norms = [norm(diff(:,:,1),2), norm(diff(:,:,2),2), norm(diff(:,:,3),2)]
norms =

   2.5573   2.6435   2.7655
```

Med 90% av detaljkoefficienterna strukna:

```
>> compressed = compressRGB(img,0.9);
>> diff = img - compressed;
>> norms = [norm(diff(:,:,1),2), norm(diff(:,:,2),2), norm(diff(:,:,3),2)]
norms =

   6.2288   5.8670   6.2970
```

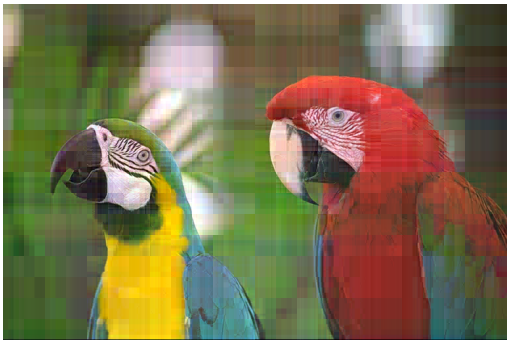
Med 95% av detaljkoefficienterna strukna:

```
>> compressed = compressRGB(img,0.95);
>> diff = img - compressed;
>> norms = [norm(diff(:,:,1),2), norm(diff(:,:,2),2), norm(diff(:,:,3),2)]
norms =

   12.945   15.130   12.955
```

(a) Originalbilden

(b) Komprimeringsgrad: 75%.

(c) Komprimeringsgrad: 90%.



(d) Komprimeringsgrad: 95%.

(e) Komprimeringsgrad: 99%.

Figur 3.4: Bilden på de två papegojorna med olika grad av komprimering. Komprimeringsgraden anger en hur stor andel av detaljkoefficienterna som har satts till 0. De koefficienter som efter transformen redan var 0 räknas inte med.

Med 99% av detaljkoefficienterna strukna:

```
>> compressed = compressRGB(img,0.99);
>> diff = img - compressed;
>> norms = [norm(diff(:,:,1),2), norm(diff(:,:,2),2), norm(diff(:,:,3),2)]
norms =

   37.222   39.067   43.071
```

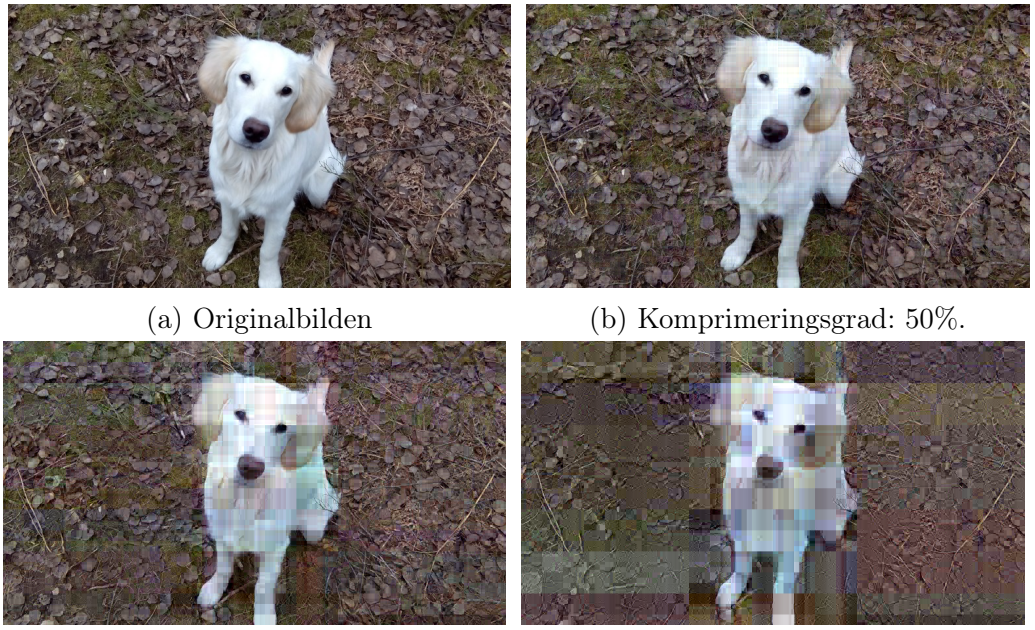
Motsvarande beräkningar för bilden på hunden ser ut så här.

```
>> img = imread("capo2.jpg");
>> img = im2double(img);
>> compressed = compressRGB(img,0.5);
>> diff = img - compressed;
>> norms = [norm(diff(:,:,1),2), norm(diff(:,:,2),2), norm(diff(:,:,3),2)]
norms =

   7.9374   8.0036   6.8613

>> compressed = compressRGB(img,0.75);
>> diff = img - compressed;
>> norms = [norm(diff(:,:,1),2), norm(diff(:,:,2),2), norm(diff(:,:,3),2)]
norms =

   15.401   14.962   15.409

>> compressed = compressRGB(img,0.90);
>> diff = img - compressed;
>> norms = [norm(diff(:,:,1),2), norm(diff(:,:,2),2), norm(diff(:,:,3),2)]
norms =

   26.556   30.446   32.341
```

Här kan noteras att matrisnormerna för t.ex. bilderna 3.5b och 3.4c är rätt så nära varandra och att de här bilderna har ungefär samma grad av artefakter från komprimeringen. Den här normen kan alltså användas istället för manuell inspektion åtminstone i vissa fall.

Beroende på vilka krav som ställs på den komprimerade signalen kan det vara nödvändigt att använda andra normer. Normen i exemplet ovan kan tillåta stora avvikelser så länge som de hålls inom ett litet område. Om sådana avvikelser inte kan tillåtas i någon enda punkt kan istället max-normen användas eftersom den mäter största avvikelsen bland alla element.

Det här avslutar delen om bildkomprimering. I följande kapitel kommer den allmänna matematiska teorin bakom krusningarna att presenteras.



(a) Originalbilden

(b) Komprimeringsgrad: 50%.

(c) Komprimeringsgrad: 75%.

(d) Komprimeringsgrad: 90%.

Figur 3.5: Den här bilden lämpar sig inte speciellt bra för komprimering med Haarkrusningar. (Privat fotografi. 2015)

Kapitel 4

Allmän teori

I det här kapitlet presenteras den allmänna teorin för krusningar samtidigt som Haarsystemet visas uppfylla de krav som ställs.

Definition 4.1. En krusning är en funktion $\psi(t) \in L^2(\mathbb{R})$ sådan att familjen

$$\psi_{j,k} = 2^{\frac{2}{j}}\psi(2^j t - k),$$

med $j, k \in \mathbb{Z}$, utgör en ortonormerad bas i Hilbertrummet $L^2(\mathbb{R})$.

Som tidigare visats kan krusningarna användas till att dela upp en signal i olika upplösningsnivåer, vilket motiverar definitionen av en *multiresolutionsanalys*.

Definition 4.2. En följd underrum $\{V_j\}_{j\in\mathbb{Z}}$ till $L^2(\mathbb{R})$ kallas för en *multiresolutionsanalys* om följande krav är uppfyllda.

1. $\cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \ldots$
2. $\bigcup_{j\in\mathbb{Z}} V_j$ ligger tätt i $L^2(\mathbb{R})$
3. $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$
4. $f(x) \in V_j \Leftrightarrow f(2^{-j}x) \in V_0$
5. $f \in V_0 \Leftrightarrow f(x-m) \in V_0$ för alla $m \in \mathbb{Z}$
6. Det existerar en funktion $\phi \in V_0$, kallad skalningsfunktion, sådan att $\{\phi(x-m)\}_{m\in\mathbb{Z}}$ utgör en bas i V_0.

Som hjälpmedel i kommande bevis kommer *enkla funktioner* att användas. Många egenskaper kan visas för enkla funktioner som ett steg på vägen mot andra typer av funktioner.

Definition 4.3. En enkel funktion $\phi : X \to \mathbb{R}$ på det mätbara rummet $(X, \mathcal{A})$ är en funktion som kan skrivas på formen

$$\phi(x) = \sum_{i=1}^{N} k_i \mathbf{1}_{E_i}$$

där $k_1, \ldots, k_N \in \mathbb{R}$ och $E_1, \ldots, E_N \in \mathcal{A}$.

Definition 4.4. Låt X vara en mängd, Σ en σ-algebra över X och μ ett mått på Σ. Då är (X, Σ, μ) ett måttrum.

För mera teori kring måttrum, σ-algebror och måtteori samt bevisen till nedanstående satser se [8]. Följande tre kända satser behövs för att visa att de enkla funktionerna ligger tätt i $L^2(\mathbb{R})$. Satserna presenteras här utan bevis.

Sats 4.5 (Monotona konvergenssatsen)**.** *Låt $\{f_n : n \in \mathbb{N}\}$ vara en växande följd av mätbara (positiva) funktioner $f_n : X \to [0, \infty]$ med*

$$f(x) = \lim_{n\to\infty} f_n(x)$$

för alla $x \in X$. Då gäller

$$\lim_{n\to\infty} \int f_n d\mu = \int f d\mu.$$

Sats 4.6 (Fatous lemma)**.** *Antag att $f_n : X \to [0, \infty], n \in \mathbb{N}$ är en följd (positiva) mätbara funktioner. Då gäller*

$$\int \lim_{n\to\infty} \inf_{k\geq n} f_k d\mu \leq \lim_{n\to\infty} \inf_{k\geq n} \int f_k d\mu$$

Sats 4.7 (Dominerade konvergenssatsen)**.** *Låt $g : X \to [0, \infty]$ vara en integrerbar funktion. Om $f_n : X \to \mathbb{R}$, med $n \in \mathbb{N}$, är en följd mätbara funktioner sådana att $\lim_{n\to\infty} f_n(x) = f(x)$ för varje $x \in X$ och $|f_n(x)| \leq g(x)$ för varje $x \in X$ så gäller*

$$\lim_{n\to\infty} \int f_n d\mu = \int f d\mu$$

Med hjälp av de här tre satserna och följande lemma går det att ställa upp en sats som bevisar att de enkla funktionerna ligger tätt i $L^2(\mathbb{R})$.

Lemma 4.8. *Antag att $f \in L^2(\mathbb{R})$ är en positiv funktion. Då existerar en växande följd av positiva enkla funktioner $\phi_n : \mathbb{R} \to [0, \infty)$ sådana att $\lim_{n\to\infty} \phi_n(x) = f(x)$ för alla $x \in \mathbb{R}$.*

Bevis. Låt $n \in \mathbb{N}$ och dela in intervallet $[0, 2^n]$ i 2^{2n} dyadiska intervall

$$I_{n,k} = (2^{-n}k, 2^{-n}(k+1)], \quad k = 0, 1, \ldots, 2^{n-1}.$$

- Visa att det existerar en följd ϕ_n som uppfyller kraven i satsen.

(a) $f \in L^2(\mathbb{R})$

(b) f är en positiv funktion.

(1) $\left\{ \begin{array}{l} \text{För varje } n \in \mathbb{N} \text{ kan vi dela in } f\text{:s värdemängd i intervallen} \\ I_n \text{ och } J_n. \end{array} \right\}$

$$I_n = [0, 2^n]$$
$$J_n = (2^n, \infty)$$

(2) $\{I_n$ kan delas in ytterligare i dyadiska intervall $I_{n,k}.\}$

$I_{n,k} = (2^{-n}k, 2^{-n}(k+1)], \quad k = 0, 1, \ldots, 2^{2n} - 1$

(3) $\left\{ \begin{array}{l} \text{Låt } E_{n,k} \text{ och } F_n \text{ beteckna de mängder i definitionsmängden} \\ \text{som motsvarar } I_{n,k} \text{ och } J_n \text{ från observationena (1) och (2). Se} \\ \text{figur 4.1.} \end{array} \right\}$

$$E_{n,k} = f^{-1}(I_{n,k})$$
$$F_n = f^{-1}(J_n)$$

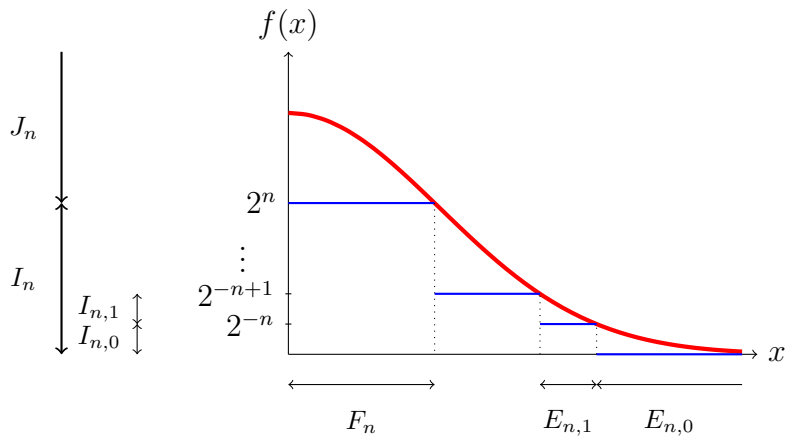
(4) $\left\{ \begin{array}{l} \text{Vi skapar en växande följd } \phi_n \text{ med hjälp av indikatorfunktio-} \\ \text{nen.} \end{array} \right\}$

$$\phi_n = \sum_{k=0}^{2^{2n}-1} 2^{-n}k\mathbf{1}_{E_{n,k}} + 2^n\mathbf{1}_{F_n}$$

$\Vdash$ $\left\{ \begin{array}{l} \text{Följden } \phi_n \text{ i (4) består av positiva enkla funktioner och} \\ \lim_{n\to\infty} \phi_n(x) = f(x) \text{ för alla } x \in \mathbb{R}. \end{array} \right\}$

□

Sats 4.9. *De enkla funktionerna ligger tätt i* $L^2(\mathbb{R})$.



Figur 4.1: Approximation med hjälp av enkla funktioner. Då n går mot ∞ blir approximationen godtyckligt bra.

Bevis. Det räcker att visa att en positiv funktion $f : \mathbb{R} \to [0, \infty)$ kan approximeras godtyckligt bra med enkla funktioner eftersom en funktion alltid kan delas upp i positiva och negativa delar, dvs. f kan skrivas som $f = f^+ - f^-$, där $f^+(x) = max\{f(x), 0\}$ och $f^-(x) = -min\{f(x), 0\}$. Alltså f^+ och f^- är icke-negativa funktioner.

Skillnaden mellan approximationen ϕ_n och f mäts med den inducerade normen i $L^2(\mathbb{R})$. Vi vill alltså visa att det för följden $\{\phi_n\}_n$ gäller att $\lim_{n\to\infty} \int_{\mathbb{R}} |f(x) - \phi_n(x)|^2 dx = 0$.

- Visa att $\lim\limits_{n\to\infty} \int_{\mathbb{R}} |f(x) - \phi_n(x)|^2 dx = 0$ då

(a) $f \in L^2(\mathbb{R})$ är en positiv funktion.

(1) $\left\{ \begin{array}{l} \text{Enligt lemma 4.8 existerar en växande följd av enkla funk-} \\ \text{tioner med det punktvisa gränsvärdet } f. \text{ Låt } \phi_n \in L^2(\mathbb{R}) \\ \text{vara en sådan följd. Nu gäller att } |f(x) - \phi_n(x)|^2 \leq (|f(x)| + \\ |\phi_n(x)|)^2 \leq 4|f(x)|^2 \text{ för alla } x \in \mathbb{R} \end{array} \right\}$

$|f(x) - \phi_n(x)|^2 \leq 4|f(x)|^2$ för alla $x \in \mathbb{R}$

(2) {Kombinera dominerade konvergessatsen med (1).}

- $\lim\limits_{n\to\infty} \int_{\mathbb{R}} |f(x) - \phi_n(x)|^2 dx$

$= \left\{ \begin{array}{l} \text{Dominerade konvergenssatsen kan nu tillämpas efter-} \\ \text{som } |f - \phi_n|^2 \leq 4|f|^2 \text{ och } 4|f|^2 \text{ är en integrerbar funk-} \\ \text{tion (eftersom } f \in L^2(\mathbb{R})). \end{array} \right\}$

$$
\begin{aligned}
& \int_{\mathbb{R}} |f(x) - f(x)|^2 dx \\
= \quad & \{|f(x) - f(x)|^2 = 0\} \\
& 0
\end{aligned}
$$

$$\lim_{n\to\infty} \int_{\mathbb{R}} |f(x) - \phi_n(x)|^2 dx = 0$$

$\Vdash$ {Resultatet följer direkt av (2).}

■ □

De enkla funktionerna är stegfunktioner och hör alltså till följden av approximationsrum V_j, från det första kapitlet, med Haarbasen som ortonormerad bas. Följande sats visar att följden av approximationsrum bildar en multiresolutionsanalys.

Sats 4.10. *Följden $\{V_j\}_{j\in\mathbb{Z}}$ av approximationsrum är en multiresolutionsanalys.*

Bevis. För att $\{V_j\}_{j\in\mathbb{Z}}$ ska vara en multiresolutionsanalys krävs att de sex egenskaperna i definition 4.2 är uppfyllda.

Egenskap 1: Den första egenskapen gäller eftersom varje dyadiskt intervall är helt inneslutet i ett dyadiskt intervall med lägre skala enligt lemma 1.6. De områden där de dyadiska funktionerna är konstanta innesluts alltså i varandra vilket innebär att en funktion i V_j också måste tillhöra V_{j+1} och således har vi $\cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \ldots$.

Egenskap 2: Enligt sats 4.9 ligger de enkla funktionerna tätt i $L^2(\mathbb{R})$ så det räcker att visa att en godtycklig enkel funktion kan approximeras med någon funktion $g \in \bigcup_{j\in\mathbb{Z}} V_j$.

- Visa att en enkel funktion f alltid kan approximeras med $g \in \bigcup_{j\in\mathbb{Z}} V_j$.

(1) $\left\{ \begin{array}{l} \text{Alla enkla funktioner kan skrivas som linjära kombinationer} \\ \text{av indikatorfunktioner.} \end{array} \right\}$

$$f(x) = \sum_{i=1}^{N} k_i \mathbf{1}_{a_i,b_i}$$

(2) $\left\{ \begin{array}{l} \text{Låt } h(x) = \mathbf{1}_{[a,b]} \text{ och antag att } a = m/2^j - a_1 \text{ och } b = n/2^j - b_1 \\ \text{samt att } a_1, b_1 < 1/2^j, \text{ där } m, n \text{ och } j \text{ är heltal. Funktionen} \\ g(x) = \mathbf{1}_{[m/2^j, n/2^j]} \text{ tillhör } \bigcup_{j\in\mathbb{Z}} V_j. \end{array} \right\}$

$$\|h-g\|_2^2 = \int_{-\infty}^{\infty} (h(x)-g(x))^2\,dx \leq 2/2^j$$

(3) $\left\{ \begin{array}{l} \text{Eftersom } j \text{ kan väljas fritt i (2) går det att göra } \|h-g\|_2^2 \\ \text{godtyckligt litet.} \end{array} \right\}$

$f(x) = \mathbf{1}_{[a,b]}$ kan approximeras med $g \in \bigcup_{j\in\mathbb{Z}} V_j$

$\Vdash$ $\left\{ \begin{array}{l} \text{Enligt observation (1) kan } f(x) \text{ skrivas som en linjär kom-} \\ \text{bination av funktioner } h_i(x) = \mathbf{1}_{[a_i,b_i]}. \text{ Varje sådan funktion} \\ \text{kan approximeras av någon funktion } g_i \in \bigcup_{j\in\mathbb{Z}} V_j \text{ enligt (3).} \\ \text{Således kan varje } f(x) \text{ approximeras med } g \in \bigcup_{j\in\mathbb{Z}} V_j. \end{array} \right\}$

■

Egenskap 3: En funktion f som tillhör $\bigcap_{j\in\mathbb{Z}} V_j$ måste vara konstant på godtyckligt stora intervall. Alltså måste f vara konstant nästan överallt. Eftersom $f \in L^2(\mathbb{R})$ innebär det här att $f(x) \equiv 0$. Vi har alltså $\bigcap_{j\in\mathbb{Z}} V_j \subseteq \{0\}$. Det är klart att $\{0\} \subseteq \bigcap_{j\in\mathbb{Z}} V_j$, alltså gäller $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$.

Egenskap 4: Varje $f \in V_j$ är konstant på de dyadiska intervallen med motsvarande skala $I_{j,k}$. Enligt definition 1.2 är $I_{j,k} = [2^{-j}k, 2^{-j}(k+1))$, dvs. $f(x)$ är konstant då

$$\frac{k}{2^j} \leq x < \frac{(k+1)}{2^j}.$$

För $2^{-j}x$ får vi då

$$\frac{k}{2^j} \leq 2^{-j}x < \frac{(k+1)}{2^j}$$
$$\Leftrightarrow$$
$$\frac{k}{2^0} \leq x < \frac{(k+1)}{2^0}.$$

Med andra ord är $f(2^{-j}x)$ konstant på varje intervall $I_{0,k}$ och således gäller $f(x) \in V_j \Rightarrow f(2^{-j}x) \in V_0$. Analogt gäller även $f(2^{-j}x) \in V_0 \Rightarrow f(x) \in V_j$.

Egenskap 5: Låt $f \in V_0$, då måste f vara konstant på varje intervall $I_{0,k}$ för $k \in \mathbb{Z}$. Funktionen $f(x)$ är då konstant när

$$k \leq x < k+1$$
$$\Leftrightarrow$$
$$k-m \leq x-m < k+1-m.$$

Funktionen $f(x-m)$ är alltså konstant på varje intervall $I_{0,k-m}$ för alla $m \in \mathbb{Z}$. Alltså gäller $f \in V_0 \Leftrightarrow f(x-m) \in V_0$ för alla $m \in \mathbb{Z}$.

Egenskap 6: Sats 1.11 säger att $\{p_{j,k}\}_{k=-\infty}^{\infty}$ utgör en ortonormerad bas för V_j, där $p_{j,k}(x) = 2^{\frac{j}{2}} p(2^j x - k)$. Om vi nu väljer $j = 0$ får vi skalningsfunktionen $p(x-k)$ för vilken gäller att $\{p(x-k)\}_{k\in\mathbb{Z}}$ utgör en bas i V_0. □

För att visa att Haarkrusningen verkligen är en krusning behövs ännu några resultat. Enligt definition 4.1 måste följden $\{2^{j/2}H(2^j x + k)\}_{j\in\mathbb{Z},k\in\mathbb{Z}}$ vara en ortonormal bas i $L^2(\mathbb{R})$ för att $H(x)$ ska vara en krusning.

Sats 4.11. *Följden $\{2^{j/2}H(2^j x + k)\}_{j\in\mathbb{Z},k\in\mathbb{Z}}$ är ortonormal i $L^2(\mathbb{R})$.*

Bevis. Enligt beteckningarna i kapitel 1 har vi $H_{j,k}(x) = 2^{j/2}H(2^j x + k)$. För att följden ska vara ortonormal måste det gälla att $\langle H_{j,k}, H_{j,k}\rangle = 1$ och $\langle H_{j,k}, H_{j',k'}\rangle = 0$ för $j \neq j'$ eller $k \neq k'$. Beviset till sats 1.15 ger direkt $\langle H_{j,k}, H_{j,k}\rangle = 1$ och dessutom $\langle H_{j,k}, H_{j,k'}\rangle = 0$ för $k \neq k'$. Det återstår alltså att visa att $\langle H_{j,k}, H_{j',k'}\rangle = 0$ för $j \neq j'$.

- Visa att $\langle H_{j,k}, H_{j',k'}\rangle = 0$ för $j \neq j'$.

(a) $j < j'$

(1) { Enligt lemma 1.6 ligger ett dyadiskt intervall $I_{j'}$ alltid helt inneslutet eller helt utanför ett intervall I_j, med $j < j'$. De områden där $H_{j,k}$ och $H_{j',k'}$ är olika 0 är I_j respektive $I_{j'}$. }

Vi har två olika fall: antingen är $I_j \cap I_{j'} = \{\emptyset\}$, eller så gäller $I_j \cap I_{j'} = I_{j'}$.

(2) { Inreprodukten ges av ytan under grafen och enligt (1) är det bara intervallet $I_{j'}$ som bidrar till den ytan. }

$$\langle H_{j,k}, H_{j',k'}\rangle = \int_{\frac{k'}{2^{j'}}}^{\frac{k'+1}{2^{j'}}} H_{j,k}(x)H_{j',k'}(x)dx$$

(3) { Om $I_j \cap I_{j'} = \{\emptyset\}$ är $H_{j,k}(x)H_{j',k'}(x) = 0$ och således gäller $\langle H_{j,k}, H_{j',k'}\rangle = 0$. Om intervallen däremot överlappar ligger $I_{j'}$ antingen i den vänstra eller högra halvan av I_j enligt beviset till lemma 1.6. Vi kan dela upp integralen i (2) i tre olika fall med hjälp av lemma 1.14. }

$$\langle H_{j,k}, H_{j',k'}\rangle = \begin{cases} 2^{\frac{j}{2}} \displaystyle\int_{\frac{k'}{2^{j'}}}^{\frac{k'+1}{2^{j'}}} H_{j',k'}(x)dx, & \text{då } I_{j',k'} \subseteq I_{j+1,2k} \\ -2^{\frac{j}{2}} \displaystyle\int_{\frac{k'}{2^{j'}}}^{\frac{k'+1}{2^{j'}}} H_{j',k'}(x)dx, & \text{då } I_{j',k'} \subseteq I_{j+1,2k+1} \\ 0, & \text{annars.} \end{cases}$$

(4) {Beräkning av integralerna i (3).}

- $\pm 2^{\frac{j}{2}} \int_{\frac{k'}{2^{j'}}}^{\frac{k'+1}{2^{j'}}} H_{j',k'}(x)dx$

$=$ {Dela upp integralen med hjälp av lemma 1.14.}

$\pm 2^{\frac{j}{2}} \left(\int_{\frac{k'}{2^{j'}}}^{\frac{k'+\frac{1}{2}}{2^{j'}}} 2^{\frac{j'}{2}} dx + \int_{\frac{k'+\frac{1}{2}}{2^{j'}}}^{\frac{k'+1}{2^{j'}}} -2^{\frac{j'}{2}} dx \right)$

$=$ {Utför integralerna.}

$\pm 2^{\frac{j}{2}} \left(2^{\frac{j'}{2}} \left(\frac{k'+\frac{1}{2}}{2^{j'}} - \frac{k'}{2^{j'}} \right) + -2^{\frac{j'}{2}} \left(\frac{k'+1}{2^{j'}} - \frac{k'+\frac{1}{2}}{2^{j'}} \right) \right)$

$=$ {Genom att addera ihop termerna får vi $\pm 2^{\frac{j}{2}} \cdot 0$.}

0

$\pm 2^{\frac{j}{2}} \int_{\frac{k'}{2^{j'}}}^{\frac{k'+1}{2^{j'}}} H_{j',k'}(x)dx = 0$

$\Vdash$ $\langle H_{j,k}, H_{j',k'} \rangle$

$=$ {Observationerna (3) och (4) ger direkt resultatet.}

0

■ □

Följden $\{2^{j/2}H(2^j x+k)\}_{j\in\mathbb{Z},k\in\mathbb{Z}}$ är alltså ortonormal i $L^2(\mathbb{R})$. För att visa att följden även är en bas behövs ännu ett hjälpresultat. Låt $S_n \subseteq L^2(\mathbb{R})$ vara det slutna underrum som ges av

$$S_n = \overline{span}\{H_{j,k}\}_{j<n,k\in\mathbb{Z}}.$$

Då gäller klart följande resultat

$$\cdots \subseteq S_{-1} \subseteq S_0 \subseteq S_1 \ldots \tag{4.1}$$

$$f(t) \in S_n \Leftrightarrow f(2t) \in S_{n+1} \tag{4.2}$$

$$f(t) \in S_0 \Leftrightarrow f(t+k) \in S_0, k \in \mathbb{Z}. \tag{4.3}$$

De här resultaten gäller också för V_j enligt sats 4.10.

Lemma 4.12. *För alla $n \in \mathbb{Z}$ gäller att $S_n = V_n$.*

Bevis. Enligt ekvation 4.2 räcker det med att visa att $S_0 = V_0$.

- Visa att $S_0 = V_0$.

(1) {Varje $H_{j,k}$ med $j < 0$ är konstant på varje intervall $[r, r+1], r \in \mathbb{Z}$.}

$S_0 \subseteq V_0$

(2) { Varje $f \in V_0$ kan skrivas som $\sum_{r\in\mathbb{Z}} k_i \mathbf{1}_{[r,r+1]}$. Om nu $\mathbf{1}_{[0,1]} \in S_0$ har vi enligt ekvation 4.3 $V_0 \subseteq S_0$. }

$\mathbf{1}_{[0,1]} \in S_0 \Rightarrow V_0 \subseteq S_0$

(3) { Serien $\sum_{j=-\infty}^{-1} 2^{j/2} H_{j,0}$ är absolut konvergent i $L^2(\mathbb{R})$ eftersom $\|2^{j/2} H_{j,0}\|_2 = 2^{j/2}$ och $j < 0$. Dessutom gäller $H_{j,0}(t) = 0$ för $t \leq 0$. }

$$\sum_{j=-\infty}^{-1} 2^{j/2} H_{j,0}(t) = 0, \text{ då } t \leq 0$$

(4) { Då $0 < t < 1$ och $j < 0$ är $H_{j,0}(t) = 2^{j/2}$ enligt definitionen på $H_{j,k}$ och $H(t)$. }

$$\sum_{j=-\infty}^{-1} 2^{j/2} H_{j,0}(t) = \sum_{j=1}^{\infty} 2^{-j} = 1, \text{ då } 0 < t < 1$$

(5) { Beräkna summan i observation (3) och (4) för $2^r < t < 2^{r+1}$, då $r = 0, 1, 2, \ldots$. }

- Beräkna $\sum_{j=-\infty}^{-1} 2^{j/2} H_{j,0}(t)$, för $2^r < t < 2^{r+1}$, då $r = 0, 1, 2, \ldots$.

(5.1) { Då $j \leq r$ har vi $2^{-j} t > 1$, eftersom $2^r < t < 2^{r+1}$, och således är $H_{-j,0}(t) = 0$. }

$$\sum_{j=1}^{r} 2^{-j/2} H_{-j,0}(t) = 0$$

(5.2) { Då $j = r+1$ har vi $2^{-j} t \in [\frac{1}{2}, 1)$, eftersom $2^r < t < 2^{r+1}$, och således är $H_{-j,0}(t) = 2^{-(r+1)/2} \cdot (-1)$. }

$2^{-j/2} H_{-j,0}(t) = -2^{-r-1}$, då $j = r+1$

(5.3) $\left\{ \begin{array}{l} \text{Då } j \geq r+2 \text{ har vi } 2^{-j}t \in [0, \frac{1}{2}), \text{ eftersom } 2^r < t < \\ 2^{r+1}, \text{ och således är } H_{-j,0}(t) = 2^{-j/2}. \end{array} \right\}$

$$\sum_{j=r+2}^{\infty} 2^{-j/2} H_{-j,0}(t) = \sum_{j=r+2}^{\infty} 2^{-j}$$

$\Vdash$ $\displaystyle\sum_{j=-\infty}^{-1} 2^{j/2} H_{j,0}(t)$

$=$ {Ändra indexeringen.}

$\displaystyle\sum_{j=1}^{\infty} 2^{-j/2} H_{-j,0}(t)$

$=$ $\left\{ \begin{array}{l} \text{Dela upp summan i tre olika delar enligt observation-} \\ \text{erna (5.1), (5.2) och (5.3).} \end{array} \right\}$

$\displaystyle 0 - 2^{-r-1} + \sum_{j=r+2}^{\infty} 2^{-j}$, då $2^r < t < 2^{r+1}$

$=$ {Den geometriska serien har summan 2^{-r-1}.}

0, då $2^r < t < 2^{r+1}$

$$\sum_{j=-\infty}^{-1} 2^{j/2} H_{j,0}(t) = 0, \text{ då } 2^r < t < 2^{r+1} \text{ och } r = 0, 1, 2, \ldots$$

$\Vdash$ $\left\{ \begin{array}{l} \text{Enligt observation (1) har vi } S_0 \subseteq V_0. \text{ Observationerna (3),} \\ \text{(4) och (5) ger tillsammans att } \mathbf{1}_{[0,1]} \in S_0. \text{ Enligt (2) gäller} \\ \text{då } V_0 \subseteq S_0 \text{ och vi får resultatet } S_0 = V_0. \end{array} \right\}$

■ □

Sats 4.11 och lemma 4.12 i kombination med egenskap 2 i sats 4.10 ger nu direkt följande sats:

Sats 4.13. *Följden* $\{2^{j/2} H(2^j x + k)\}_{j \in \mathbb{Z}, k \in \mathbb{Z}}$ *är en ortonormal bas i* $L^2(\mathbb{R})$.

Med det här resultatet står det nu klart att Haarkrusningen $H(x)$ mycket riktigt är en krusning.

Litteratur

[1] P. Wojtaszczyk. *A Mathematical Introduction to Wavelets*. London Mathematical Society Student Texts. Cambridge University Press, 1997. ISBN: 9780521578943.

[2] L. Bengtsson och B. Karlström. *Transformer - från jω till Wavelets*. Studentlitteratur, 2007. ISBN: 9789144047010.

[3] E.J. Stollnitz, T.D. DeRose och D.H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann series in computer graphics and geometric modeling. Morgan Kaufmann Publishers, 1996. ISBN: 9781558603752.

[4] M. Frazier. *An Introduction to Wavelets Through Linear Algebra*. Undergraduate Texts in Mathematics. Springer, 1999. ISBN: 9780387986395.

[5] S. Axler. *Linear Algebra Done Right*. Undergraduate Texts in Mathematics. Springer New York, 1997. ISBN: 9780387982595.

[6] N. Young. *An Introduction to Hilbert Space*. Cambridge mathematical textbooks. Cambridge University Press, 1988. ISBN: 9780521337175.

[7] Eric W. Weisstein. *Matrix Norm. From MathWorld–A Wolfram Web Resource*. URL: `http://mathworld.wolfram.com/MatrixNorm.html` (hämtad 2015-07-07).

[8] W. Rudin. *Real and complex analysis*. Mathematics series. McGraw-Hill, 1987. ISBN: 9780070542341.

[9] I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial och Applied Mathematics, 1992. ISBN: 9780898712742.

[10] John W. Eaton m. fl. *GNU Octave*. Version 3.8.1. 2015. URL: `http://www.octave.org`.

[11] *Nationalencyklopedin NE.se*. Malmö, 2000.

[12] J.J. O'Connor och E.F. Robertson. *Alfréd Haar. Biography.* URL: `http://www-history.mcs.st-andrews.ac.uk/Biographies/Haar.html` (hämtad 2016-06-30).

[13] Ralph-Johan Back. *Structured Derivations as a Unified Proof Format for Teaching Mathematics.* Tekn. rapport 949. TUCS, juli 2009.

Bilaga A

Programkod för Octave

I den här bilagan finns implementationer för några funktioner som kan användas för bildkomprimering i Octave.

Exempelkod A.1: Funktionen `wave`.

```
function [ret, appr, detail] = wave(x, num)
  % Transforms the vector x num times.
  % ret = the vector transformed in place
  % appr = the approximation given by the averages
  % detail = the detail coefficients
  detail = [];
  for n = 1:num % transform num times
    % number of columns is halved each time
    for i = 0:columns(x)/pow2(n)-1
      % detail coefficients: [a, b] -> [a, (a-b)/2]
      x(:,pow2(n)*i+1+pow2(n-1)) = (x(:,pow2(n)*i+1) - x(:,pow2(n)*i+1+pow2(n-1)))/2;
      % avrages: [a, (a-b)/2] -> [(a+b)/2, (a-b)/2]
      x(:,pow2(n)*i+1) = x(:,pow2(n)*i+1) - x(:,pow2(n)*i+1+pow2(n-1));
    endfor
    detail = [x(:,linspace(1+pow2(n-1),columns(x)-pow2(n-1)+1,columns(x)/pow2(n))),
        detail];
  endfor
  ret = x;
  appr = x(:,linspace(1, columns(x)-pow2(num)+1, columns(x)/pow2(num)));
endfunction
```

Exempelkod A.2: Funktionen `unwave`.

```
function ret = unwave(x, num)
  %x is the vector of values [(a+b)/2, b-a], num is the number of transforms
  for n = num-1:-1:0 % transform num times
    for i = 0:columns(x)/pow2(n+1)-1
      % calculate a from [(a+b)/2, (a-b)/2]
      x(:,pow2(n+1)*i+1) = x(:,pow2(n+1)*i+1) + x(:,pow2(n+1)*i+1+pow2(n));
      % calculate b from [a, (a-b)/2]
      x(:,pow2(n+1)*i+1+pow2(n)) = x(:,pow2(n+1)*i+1) - 2*x(:,pow2(n+1)*i+1+pow2(n));
    endfor
  endfor
  ret = x;
endfunction
```

Exempelkod A.3: Funktionen `decompose`.

```
function [result, coarser, detail] = decompose(img)
  num = log(columns(img))/log(2);
  % Transform rows.
  [result, coarser, detail] = wave(img, num);
  % Transpose
  coarser = result';
  num = log(columns(coarser))/log(2);
  % Transform columns.
  [ret, coarser, detail] = wave(coarser, num);
  % Transpose back to original orientation.
  result = ret'; coarser = coarser'; detail = detail';
endfunction
```

Exempelkod A.4: Funktionen `recompose`.

```
function result = recompose(img)
  % Number of transforms
  num = log(columns(img))/log(2);
  % Transpose image first to unwave columns.
  img = img';
  result = unwave(img, num);
  % Back to normal rotation.
  img = result';
  result = unwave(img, num);
endfunction
```

Exempelkod A.5: Funktionen `compress`.

```
function result = compress(img, fact)
  % This function takes the decomposed image img
  % and compresses it by removing fact*total of
  % the detail coefficients.

  % Total numer of non zero elements.
  total = nnz(img);
  % get rid of zeros since we are looking for smallest absolute value that is non zero
  img(~img) = inf;

  % The compression factor. We are going to remove compFactor*total values.
  compFactor = fact;
  str = sprintf('Compression␣factor:␣%d%%',compFactor*100);
  disp(str);
  % measuring time and displaying loading bar
  barhandle = waitbar(0,'Compressing...');
  tic();

  % get the min values of every column and the index in the column for every value
  % colind = [index of first rows min, ind of second rows min...]
  [values, colind] = min(abs(img));
  % the global minimum value and index of the column where it is
  [minval, ind] = min(values);

  for i = 1:round(compFactor*total)
    % This is not needed the first time through the loop.
    % Get a new min of the column that had the global minimum.
    [values(ind), colind(ind)] = min(abs(img(:,ind)));
    % Get global minimum
    [minval, ind] = min(values);

    % remove the minimum
    img(colind(ind), ind) = inf;

    % update progress
    if (mod(i, 5000) == 0)
      progress = i/(compFactor*total);
      waitbar(progress, barhandle);
    endif
  endfor

  img(isinf(img)) = 0; % put back the zeros
  toc();
  close(barhandle);
  result = img;
endfunction
```

Exempelkod A.6: En funktion för att komprimera färgbilder.

```
function compressed = compressRGB(img, fact)
  img = im2double(img);
  % Split into R,G and B
  imgr = img(:,:,1); imgg = img(:,:,2); imgb = img(:,:,3);
  % Compression
  compr = compressImage(imgr, fact);
  compg = compressImage(imgg, fact);
  compb = compressImage(imgb, fact);
  % Put back together
  compressed(:,:,1) = compr;
  compressed(:,:,2) = compg;
  compressed(:,:,3) = compb;
  % Fix rounding errors
  compressed = im2double(im2uint8(compressed));
endfunction
```

Exempelkod A.7: Funktionen `compressImage`.

```
function [compressed, decomposed] = compressImage(img, fact)
  % Compress the image img using the compression factor fact.

  % Add padding if needed.
  % bottom padding
  ex = ceil(log(rows(img))/log(2)); % exponent
  paddingBottom = zeros(pow2(ex) - rows(img), columns(img));
  img = [img; paddingBottom];

  % padding on the side
  ex = ceil(log(columns(img))/log(2));
  paddingSide = zeros(rows(img), pow2(ex) - columns(img));
  img = [img, paddingSide];

  % Decompose image
  [img, appr, detail] = decompose(img);

  % Zero out appr in img to protect it in the compression
  num = log(rows(img))/log(2);
  img(1,:) = 0;

  % Compress the decomposed image
  img = compress(img,fact);

  % Put back appr
  img(1,:) = appr;

  % Decomposed image without padding
  decomposed = img(:,1:end-columns(paddingSide));
  decomposed = decomposed(1:end-rows(paddingBottom),:);

  % Recompose the image
  compressed = recompose(img);

  % Remove padding
  compressed = compressed(:,1:end-columns(paddingSide));
  compressed = compressed(1:end-rows(paddingBottom),:);
endfunction
```