Opgave Project Datacommunicatie

Samuel Van de Velde
samuel.vandevelde@telin.ugent.be

5 oktober 2012

1 Inleiding

Dit is de opgave voor het project dat aansluit bij de cursus *Datacommunicatie*. Dit project wordt uitgevoerd in groepjes van 3 of 4 studenten. Inschrijven in een groep gebeurt via Minerva. Het groepsnummer bepaalt een aantal van de parameters die nodig zijn bij het oplossen van het project.

Het doel van dit project is het simuleren van een digitaal communicatiesysteem. Hierbij wordt uitgegaan van een simpel model voor het kanaal waarbij er voor elke bit een zekere kans bestaat dat deze foutief ontvangen wordt. Om de verzonden informatie te beschermen tegen deze eventuele fouten, maken we gebruik van een passende kanaalcodering. Verder wordt de te verzenden informatie ook gecomprimeerd met een broncoderingsalgoritme.

Paragraaf 2 geeft een overzicht van de verschillende componenten waaruit het beschouwde digitaal communicatiesysteem is opgebouwd. In paragraaf 3 wordt de opgave van het project beschreven en wordt de implementatie stap voor stap uitgelegd.

2 De componenten van het digitaal communicatiesysteem

2.1 Zender

De zender heeft als taak de digitale informatie die aangeboden wordt door een bron om te zetten in een fysisch informatiesignaal dat kan verzonden worden over het kanaal. Volgende stappen worden daarvoor uitgevoerd:

- Eerst wordt de zwart-wit afbeelding ingelezen uit een bestand. Een zwarte pixel komt overeen met bit '1' en een witte pixel met bit '0'.
- De ingelezen afbeelding bevat meestal redundantie, dit is overtollige informatie. In ons geval bestaat de figuur uit veel witte of zwarte vlakken (zie figuur 1). De te versturen sequentie van informatiebits kan dan ook worden ingekort of *gecomprimeerd* door het samen nemen van verschillende pixels in zogenaamde macrosymbolen. Men kent dan een korte bitsequentie toe aan veel voorkomende macrosymbolen en een langere bitsequentie

Figuur 1: Te versturen afbeeldingen.

aan zeldzame macrosymbolen. De omzetting van de bronuitgang naar een bitsequentie wordt *broncodering* genoemd.

- Als we informatie versturen over een kanaal, worden onvermijdelijk fouten geïntroduceerd. Om onze bitsequentie te beschermen tegen dergelijke fouten, voegen we op gecontroleerde wijze redundantie toe. Deze procedure wordt *kanaalcodering* genoemd. Hierbij wordt de bitsequentie opgesplitst in groepjes van k bits en aan elk groepje worden $(n-k)$ redundante bits toegevoegd. Op die manier verkrijgen we codewoorden van n bits. Dit leidt tot een nieuwe, langere bitsequentie.

2.2 Kanaal en ontvanger

Het kanaal dat hier beschouwd zal worden is het binair symmetrisch kanaal met parameter p. Om de ontvangen bitsequentie te reconstrueren zullen volgende twee stappen uitgevoerd moeten worden:

- de kanaaldecoder zal fouten in de ontvangen woorden corrigeren
- de afbeelding wordt gedecomprimeerd.

3 Opgave

Het doel van dit project is het implementeren van voorgaande stappen en het onderzoeken en bespreken van een aantal facetten van het communicatiesysteem. De verschillende componenten worden eerst afzonderlijk onderzocht in paragrafen 3.1 en 3.2. Uiteindelijk worden beide componenten samen getest in paragraaf 3.3. De te gebruiken afbeeldingen wordt ter beschikking gesteld via minerva, evenals een aantal Matlab-functies die nuttig kunnen zijn bij het oplossen van het project.

Bij dit project dient een verslag te worden opgesteld waarin de onderstaande vragen worden beantwoord en de resultaten worden geïllustreerd. **Dit afgedrukte verslag moet ingediend worden ten laatste op 21 december 2012[1].** De MATLAB code wordt ingeleverd via minerva als zipbestand met naam *groepXX.zip* met daarin een map *groepXX* waarin alle code staat.

3.1 Broncodering

Elke groep zal een grafisch bestand toegekend krijgen om te comprimeren. Voor de compressie maken we gebruik van Huffmancodering. Hierbij gaan we als volgt te werk. Het grafisch bestand kan geïnterpreteerd worden als een matrix waarbij de dimensies overeenstemmen met het aantal pixels op een horizontale lijn en verticale lijn in de figuur. Elk element uit deze matrix kan de waarde 0 of 1 aannemen corresponderend met een zwarte pixel of een witte pixel. We gaan de bitmap opsplitsen in blokken van 2 bij 2 pixels. In het bestand *Source_ coding.m* zal je de functies vinden die je moet implementeren.

1. Geef de relatieve frequenties van de macrosymbolen van de afbeelding. Stel aan de hand van deze relatieve frequenties handmatig de Huffmancode op. Geef de boom weer en de resulterende code. Geef het gemiddeld aantal codebits $\mathbb{E}[n]$ per bronsymbool voor deze code.
2. Implementeer de functie *create_ codebook()* die de Huffmancode opstelt voor gegeven symbolen en relatieve frequenties.

Om een beter inzicht te krijgen in het concept van entropie gaan we de data in het bestand *'Xfile.mat'* comprimeren.

3. Geef de entropie voor de macrosymbolen weer in een grafiek voor $K = 1..10$ waarbij K het aantal bits is dat je samenneemt in een macrosymbool. Geef ter referentie ook de entropie weer voor macrosymbolen die allen even waarschijnlijk zijn. Wat is de algemene trend voor stijgende K? Verklaar.
4. Bereken het gemiddeld aantal codebits $\mathbb{E}[n]$ per bronsymbool gebruikmakend van Huffmancodering voor $K = 1..10$. Geef de waarde $\mathbb{E}[n]$ weer in een grafiek samen met de boven- en ondergrens voor $\mathbb{E}[n]$, voor de verschillende waarden van K. Hoeveel bits neem je samen om het bestand optimaal te comprimeren? Verklaar je antwoord en geef de resulterende compressiefactor.
5. Komt $\mathbb{E}[n]$ dicht bij de ondergrens? Wat kunnen we hieruit concluderen? Is het mogelijk om met Huffmancodering exact de ondergrens te bereiken? Onder welke voorwaarden kan dit gebeuren?

[1] Technicum verdieping -T, de gang volgen en de trap naar boven nemen. Afgeven aan Samuel Van de Velde of aan het secretariaat.

$c_{1,1}$	$c_{1,2}$	$\cdots$	$c_{1,n}$
$c_{2,1}$	$c_{2,2}$	$\cdots$	$c_{2,n}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$c_{l,1}$	$c_{l,2}$	$\cdots$	$c_{l,n}$
p_1	p_2	$\cdots$	p_n

Tabel 1: Toevoegen van extra pariteitsbits

Om het bestand aan de ontvangerzijde te kunnen decomprimeren hebben we het woordenboek nodig. In appendix A staat een methode beschreven waarbij de Huffmancode wordt omgezet in een canonische vorm zodat het corresponderende woordenboek efficiënt kan worden voorgesteld. Lees de appendix en los volgende vragen op:

6. Geef de Canonische Huffmancode voor de afbeelding uit de vorige sectie.

7. Gegeven is het alfabet {A, B, C, D, E, F, G} waarbij de lengtes van de codewoorden respectievelijk 5, 3, 3, 1, 4, 5 en 3 zijn. Geef de codewoorden in canonische vorm.

8. Implementeer de functie *create_ canonical_ codebook()* die aan de hand van het alfabet en codelengtes de canonische Huffmancode opstelt.

3.2 Kanaalcodering

In dit deel van het project werk je best met een *zelf gegenereerde random bitsequentie* in plaats van het grafisch bestand (tenzij anders vermeld). We wensen een productcode te gebruiken om de te versturen informatie te beschermen tegen eventuele fouten. In het bestand *Channel_ Coding.m* zal je de functies vinden die je moet implementeren.

1. We vertrekken van de cyclische (15,11) Hammingcode met generator veelterm $g(x)$ (wordt bepaald door groepsnr.). Wat is de minimale Hammingafstand van deze code? Wat is het foutdetecterend en foutcorrigerend vermogen van deze code? Bepaal de checkveelterm $h(x)$. Bepaal de generator- en checkmatrix van deze code. Zet de bekomen matrices om in systematische vorm: $\mathbf{G}_{sys} = (\mathbf{I}_{11} | \mathbf{P})$ en $\mathbf{H}_{sys} = \left(\mathbf{P}^T \middle| \mathbf{I}_4\right)$.

2. Bepaal de syndroomtabel aan de hand van $\mathbf{H}_{sys}$ en bespreek. Onderstel transmissie over een binair symmetrisch kanaal met bitfoutprobabiliteit p. Indien de code enkel wordt aangewend voor foutcorrectie, wat is dan de kans op een decodeerfout? Geef de exacte formule, alsook een benadering voor $p \ll 1$.

3. Implementeer nu de kanaalencoder (*Ham_ Encode()*) en de kanaaldecoder (*Ham_ Decode()*) op basis van de systematische code. Voer simulaties uit om de kans op een decodeerfout te bepalen. Beschouw een binair symmetrisch kanaal met $p = 3 \times 10^{-1},\ 1 \times 10^{-1},\ 3 \times 10^{-2},\ 1 \times 10^{-2},\ 3 \times 10^{-3},\ 1 \times 10^{-3}$. Vergelijk met het analytische resultaat uit vraag 2. Maak een figuur, als functie van de foutprobabiliteit p, van de gevonden simulatieresultaten en van het analytische resultaat voor de kans op een decodeerfout (gebruik een dubbellogaritmische schaal).

4. Om de te versturen informatie nog beter te beschermen, gaan we gebruik maken van een productcode. Dit is een foutcorrigerende code die bestaat uit het 'product' van 2 foutcorrigerende codes. Dit is een eenvoudige techniek om de minimale Hammingafstand te vergroten[2]. Als eerste code nemen we de reeds besproken Hamming code, als tweede code nemen we een code die aan elk codewoord een pariteitsbit toevoegt. De encodering gebeurt als volgt: we beschouwen l informatiewoorden die we eerst encoderen met de (15,11) Hamming code uit vraag 3, die we allemaal samenvoegen in een matrix (zie tabel 1). Aan elke kolom voegen we nog een pariteitsbit toe. De $l \times k$ informatiebits worden omgezet in $(l+1) \times n$ codebits. Deze bits worden dan rij per rij verstuurd. Wat is de minimale Hammingafstand van deze resulterende code? Neem $l = 8$. Pas nu de functie *Prod_ encode()* aan zodat hij deze productcode implementeert.

5. We gaan nu de decoder implementeren voor deze code. De decoder splitst het ontvangen codewoord op in $(l+1)$ rijen van n bits. Per rij wordt het syndroom berekend met behulp van de checkmatrix van de Hammingcode en per kolom wordt de pariteit berekend. Aan de hand van de syndromen en pariteitsbits neemt de decoder een beslissing. Algoritme 1 beschrijft het te volgen decodeerproces voor deze code. Pas de

[2]De minimale Hammingafstand van een productcode is gelijk aan het product van de minimale Hammingsafstanden van beide foutcorrigerende codes.

Algoritme 1 Het decodeerproces van de productcode

- Bereken de syndromen voor alle rijen en alle pariteitsbits voor alle kolommen
- Als alle syndromen gelijk zijn aan nul, doe niets want er is geen fout opgetreden
- Indien er toch syndromen verschillend zijn van nul, bepaal de plaats van de fout per syndroom dat verschillend is van nul
- Kijk per gevonden foutpositie of de overeenkomstige pariteitsbit gelijk is aan 1: i) Indien dit zo is, corrigeer dan de bit en zet de berekende pariteitsbit en het overeenkomstig syndroom op nul. ii) Indien dit niet zo is, betekent dit dat er meerdere fouten op dezelfde lijn voorkomen, doe voorlopig niets.
- Indien er na vorige stap nog syndromen en pariteitsbits verschillend van nul zijn, dan zijn er rijen of kolommen met meerdere fouten: i) Indien er meer dan 2 syndromen verschillend van nul zijn, doe niets. ii) Indien er slechts 1 syndroom verschillend is van nul, dan komen er twee fouten voor in die bepaalde rij. Kijk naar de pariteitsbits die verschillend zijn van nul en verbeter de overeenkomstige bits in de beschouwde rij. iii) Indien er 2 syndromen zijn die verschillen van nul en bovendien gelijk zijn aan elkaar, betekent dit dat er 2 fouten optreden in 1 kolom. Bepaal de positie van de fouten aan de hand van het syndroom en verbeter beide bits.

functienaam	*omschrijving*
imread, imshow	Functies om een grafisch bestand in te lezen en te plotten.
reshape	Functie om matrices te vervormen.
de2bi, bi2de	Functies om integers te converteren naar bits en vice versa.
unique	Vind alle unieke elementen in een vector of matrix.
sort	Sorteer een vector.

Tabel 2: Enkele nuttige MATLAB commando's

functie *Prod_ decode()* aan zodat deze het beschreven decodeerproces kan uitvoeren. Zijn er foutpatronen die de oorspronkelijke Hammingcode wel kan corrigeren, maar deze nieuwe decoder niet?

6. Voer opnieuw simulaties uit om de kans op een decodeerfout te bepalen. Beschouw een binair symmetrisch kanaal met $p = 3 \times 10^{-1},\ 1 \times 10^{-1},\ 3 \times 10^{-2},\ 1 \times 10^{-2},\ 3 \times 10^{-3},\ 1 \times 10^{-3}$. We gaan deze resultaten vergelijken met de resultaten voor de Hammingcode. Aangezien de productcode l Hammingcodewoorden per codewoord verstuurt, gaan we deze resultaten vergelijken met de kans dat er bij de Hammingcode, gedurende l codewoorden minstens 1 decodeerfout optreedt. Bepaal de analytische uitdrukking voor deze kans en vergelijk met de gevonden simulatieresultaten voor de productcode. Maak opnieuw een figuur.

7. Verstuur nu de afbeelding (zonder compressie) over een binair symmetrisch kanaal met $p = 1 \times 10^{-2}$. Beschouw volgende 3 situaties: *i*) niet gecodeerde transmissie *ii*) gecodeerde transmissie met de Hammingcode *iii*) gecodeerde transmissie met de productcode. Geef de ontvangen afbeeldingen weer en bespreek de verschillen.

3.3 Volledig systeem

Verstuur nu afbeelding 1 (met compressie in blokken van 2×2) over een binair symmetrisch kanaal met $p = 1 \times 10^{-2}$. Beschouw volgende 2 situaties: i) transmissie zonder codering en compressie ii) gecomprimeerde transmissie iii) gecomprimeerde en gecodeerde transmissie met de productcode. Geef de ontvangen afbeeldingen weer en bespreek de verschillen.

4 MATLAB implementatie

Het is belangrijk dat de functies die moeten worden aangevuld de juiste input en output kunnen verwerken. Dit staat steeds in detail uitgelegd in de meegeleverde bestanden. De functies die je zo opstuurt kunnen getest worden op de juiste werking. Let op dat alle bestanden tevens gecontroleerd worden op plagiaat. In tabel 2 worden enkele MATLAB commando's gegeven die nuttig kunnen zijn voor dit project. Zet ook voldoende commentaar bij je code.

Appendix A: Canonische Huffmancode

De set van alle macrosymbolen noemen we het alfabet en om decompressie mogelijk te maken moet voor elk symbool in het alfabet het corresponderende codewoord gekend zijn. Beschouw onderstaande Huffmancode met alfabet {A,B,C,D}:

A = 11
B = 0
C = 101
D = 100

Om een gecomprimeerd bestand met deze code te decomprimeren is voor elk symbool uit het alfabet volgende informatie nodig: de lengte van het codewoord, en het codewoord zelf.

('A',2,11), ('B',1,0), ('C',3,101), ('D',3,100)

Deze informatie wordt het woordenboek genoemd en dient mee verstuurd te worden met het gecomprimeerde bestand. Aangezien we het woordenboek in alfabetische volgorde meegeven kunnen we de letters van de macrosymbolen weglaten uit de representatie van het woordenboek.

(2,11), (1,0), (3,101), (3,100)

Door de Huffmancode om te zetten naar zijn canonische vorm wordt het mogelijk om het woordenboek te reconstrueren enkel en alleen gebruikmakende van de lengte van de codewoorden. De canonische vorm steunt op het principe dat een canonisch codewoord steeds een hogere decimale waarde heeft dan een eerder codewoord met dezelfde lengte.

Om een Canonische Huffmancode te bekomen gaat men als volgt te werk: rangschik de codewoorden *eerst* op stijgende lengte en *ten tweede* op de alfabetische waarde van het macrosymbool:

B = 0
A = 11
C = 101
D = 100

De bestaande codewoorden worden vervolgens één voor één vervangen door nieuwe codewoorden met gelijke lengte gebruikmakende van volgende regels:

- Het eerste codewoord wordt vervangen door enkel nullen.
- Elk daaropvolgend codewoord wordt vervangen door het hetzelfde binair getal als het vorige, verhoogd met 1.
- Wanneer een codewoord een grotere lengte heeft als het vorige codewoord worden er, na het verhogen, nullen toegevoegd tot wanneer het codewoord de juiste lengte heeft.

Voor de bovenstaande code geeft dit volgend resultaat:

B = 0
A = 10
C = 110
D = 111

Merk op dat in Canonische vorm, de code nog steeds een prefix-code is en dus uniek decodeerbaar is. Het woordenboek kan nu efficiënt weergegeven worden als volgt:

(2),(1),(3),(3)

Bij het decomprimeren wordt de canonische code opnieuw opgesteld door middel van deze informatie.