Nie-destruktiewe klankonttrekking, restourasie en spraakverheffing van Edison-fonograafsilinders

deur

Ewald van der Westhuizen

Tesis ingelewer ter gedeeltelike voldoening aan die vereistes vir die graad Magister in Natuurwetenskappe (Ingenieurswese) aan die Universiteit van Stellenbosch

Studieleier: Dr. M. M. Blanckenberg

Mede-studieleier: Mnr. L. Schwardt

Desember 2003

Verklaring

Ek, die ondergetekende, verklaar hiermee dat die werk in hierdie tesis vervat, my eie oorspronklike werk is en dat dit nie vantevore in die geheel of gedeeltelik by enige universiteit ter verkryging van 'n graad voorgelê is nie.

Opsomming

Twee nie-destruktiewe metodes om klank van Edison-fonograafsilinders te onttrek, is ondersoek. 'n Opneemtoestel, wat die silinders met baie hoë akkuraatheid posisioneer, is ontwerp en vervaardig. 'n Mikroskopiese beeldmetode is as eerste klankonttrekkingsmetode ondersoek. Mikroskopiese beelde is met 'n webkamera van die silinderoppervlak geneem. Klank is vanuit die wydtemodulasie sigbaar in die beelde onttrek. Resultate was nie bevredigend nie weens groefintermodulasie-eggo's en 'n tekort aan resolusie. Die ware modulasie van die klank is nie in die wydte van die groefie gegraveer nie, maar in die diepte. Die tweede klankonttrekkingsmetode gebruik 'n aangepaste lasersensor van 'n CD-speler om die dieptemodulasie van die groefie te meet. Drie laseropneemmetodes is ondersoek. Die eerste is voorwaartse opname, wat die dieptemodulasie in die opneemrigting van die groefie meet. 'n Tweede opneemmetode, truwaartse opname, is identies aan voorwaartse opname, behalwe dat die meganiese stelsel in trurat beweeg. Vier opnames vanuit verskillende posisies in die groefbreedte is gekombineer om 'n klanksein te vorm. Die kombinasie van vier opnames toon 'n beduidende verbetering op die sein-tot-ruis-verhouding. Dit het aanleiding gegee tot die derde opneemmetode, dwarsskandering, wat die hele profiel van die groef meet. Die groefprofiel word dan verwerk tot 'n klanksein. 'n Handoudiorestourasieprogram is geskryf om sigbare verwringing in die klanksein met beter interpolasies te vervang. Twee spraakverheffingsmetodes is ondersoek. Short-Time Spectral Attenuation (STSA) is die mees gebruikte metode vir oudiorestourasie. 'n Tweede spraakverheffingsmetode wat van 'n lineêre voorspellings-koëffisiëntafskatting (LPC-afskatting) van korttydraampies gebruik maak, is ook toegepas. Die twee metodes is deur luistertoetse teen mekaar opgeweeg. Die LPC-metode is verkies aangesien dit die verstaanbaarheid van die spraak beter behoue laat bly.

Abstract

Two non-destructive methods of extracting audio from Edison phonographic cylinders were investigated. A recording device with high accuracy positioning was designed and manufactured. A microscopic image method was investigated first. Surface images of the cylinder were obtained using a webcamera. An audio signal was then extracted from the width modulation. Results were not pleasing as echoes caused by intergroove modulation were perceptable. The audio also lacked resolution. The true modulation of the audio is not embedded in the width, but in the depth of the groove. The second audio extraction method involved using a laser pick-up from a compact disc player to measure the depth of the groove. Three laser recording methods were investigated. The first was forward recording, that measured the depth modulation in the recording direction of the groove. The second method, backward recording, was identical to forward recording with the mechanical system moving in reverse. Four recordings from different positions in the groove were combined to create an audio signal. This combination of recordings showed a substantial improvement in the signal-to-noise ratio. A third recording method, transverse recording, that measured the whole depth profile of the groove was also investigated. The groove profile was then processed to an audio signal. A manual audio restoration program was written to replace visible sections of distorted data with better interpolations. Two speech enhancement methods were investigated, the first being the most commonly used speech enhancement method for digital audio restoration, Short-Time Spectral Attenuation (STSA). The second method is based on linear predictive coefficient (LPC) estimation of short-time frames. The two methods were evaluated by means of listening tests. The LPC enhancement method was preferred because it enhanced the intelligibility of the speech.

Inhoudsopgawe

Bedankings

Baie dankie aan dr. Blanckenberg vir sy leiding en wyse raad. Dit was 'n voorreg om saam met u te kon werk.

Baie dankie, Ludwig, vir jou geduld en tyd.

Dankie aan my familie en vriende, vir ondersteuning, bemoediging en gebede.

Dankie aan die Langenhoven Gedenkfonds en Vriende van Arbeidsgenot. Sonder U befondsing sou hierdie projek nie moontlik gewees het nie.

My grootste dank aan die Hemelse Vader wat my hierdeur gedra het.

Lys van simbole

Hoofstuk 3

l_s	Silinderlengte (aksiaal)
l_d	Geldige datalengte (aksiaal)
d_o	Buitedeursnit van silinder
o_o	Omtrek van silinder
p_s	Steek van groef
m_d	Groef se dieptemodulasie
T_{oos}	Periode van oorspronklike opname (rotasiespoed)
f_{oos}	Frekwensie van oorspronklike opname (rotasiespoed)
$f_{monster}$	Monsterfrekwensie
s_{rot}	Rotasiestaplengte
s_{tr}	Translasiestaplengte

Hoofstuk 4

φ_L	Beligtingshoek van invallende lig
φ_K	Hoek van die kamers se optiese as
k	Viterbi-tydstapindeks
K	Aantal Viterbi-tydstappe
M	Aantal Markov-toestande
i	Indeks vir huidige Markov-toestand

j	Indeks vir volgende Markov-toestand
a_{ij}	Oorgangswaarskynlikheid van toestand i na j
$b_i(k)$	Toestandskostefunksie van toestand i by tydstap k
π_i	Gekose aanvangswaarskynlikheid vir elke toestand i
δ	Matriks met gedeeltelike uitkomste (waarskynlikhede) van Viterbi-algoritme
$\Psi_k(i)$	Matriks van mees waarskynlike toestandsoorgange
P^*	Maksimum uitkoms van die Markov-proses
i_K^*	Toestand wat die maksimum uitkoms van die Markov-proses gelewer het

Hoofstuk 5

K_{Som}	Sommeerderaanwins
K_{NN}	Naloopnetwerkaanwins
K_I	Integraalaanwins
K_D	Digitale aanwins
T	Monsterperiode van digitale integreerder
f_s	Monsterfrekwensie van die digitale integreerder
T_{op}	Omwentelingperiode van silinder tydens laseropname
f_{oos}	Frekwensie van oorspronklike opname (rotasiespoed)
ω_L	Laagafsnyfrekwensie van geslotelus lasersensor-terugvoerstelsel

ω_H	Hoogafsnyfrekwensie van geslotelus lasersensor-terugvoerstelsel
p_s	Steek van groef
ω_c	Hoogafsnyfrekwensie van die geslotelus dwarsskandering-stelsel

Hoofstuk 6

M	Aantal Markov-toestande
a_{ij}	Oorgangswaarskynlikheid van toestand i na j
$b_i(k)$	Toestandswaarskynlikheid van toestand i by tydstap k
K	Aantal Viterbi-tydstappe

Hoofstuk 7

$\underline{s}_i$	Stochastiese veranderlike van die klanksein onder restourasie
N	Lengte van korttydraampie s_k
s_k , $k = 1,...,N$	Korttydraampie van klanksein onder restourasie
m	Aantal onbekende monsters
$t(i)$, $i = 1,...,m$	Posisies van onbekende monsters
V	Versameling van die posisies van die onbekende monsters
W	Versameling van monsterposisies in die korttydraampie (bekend en onbekend)

$W \setminus V$	Versameling van die posisies van die bekend monsters
$\hat{s}_{t(i)}$, $i = 1,...,m$	Afskatting van die monsterwaardes by die posisies van die onbekende monsters
$\mathbf{R}_{(\text{bekende monsters})}$	$(N-m)\times(N-m)$ outokorrelasie matriks van die klanksein
$\mathbf{R}'$	$(N-m)\times N$ gereduseerde outokorrelasie matriks
v_i , $i = 1,...,N$	Korttydraampie waarvan die monsterwaardes by die posisies van die onbekende monsters nul is
p	Die orde van die outoregressiewe model
$\hat{\mathbf{x}} = \left[\hat{s}_{t(1)}\ \hat{s}_{t(2)} \cdots \hat{s}_{t(m)}\right]^{\mathrm{T}}$	Vektorrangskikking van die afskatting van die onbekende monsters
$y(n)$	Waargenome klanksein met ruis
$x(n)$	Oorspronklike klanksein
$d(n)$	Lukrake ruis
k	Indeks vir spektrale komponente
H_k	STSA aanpasbare filter
$\mathbf{Y}_k$	Korttyd groottespektrum van die waargenome klanksein
$\mathbf{X}_k$	Korttyd groottespektrum van die oorspronklike klanksein
$\hat{\mathbf{X}}_k$	Afskatting van die korttyd groottespektrum van die oorspronklike klanksein
$S_X(\omega)$	Relatiewe seindrywingspektrum
$S_D(\omega)$	Relatiewe ruisdrywingspektrum

$\hat{S}_D(\omega)$	Afskatting van die ruisdrywingspektrum

Afkortings

A/D	Analoog na digitaal (omsetter)
AR	Outoregressief
CCD	Charge coupled device
D/A	Digitaal na analoog (omsetter)
LED	Ligstralende diode / light emitting diode
LPC	Lineêre voorspellingskoëffisiënt
OV	Operasionele versterker
SNR	Sein-tot-ruis-verhouding
STFT	Korttyd Fourier-transform
STSA	Short-time spectral attenuation

Lys van Figure

Lys van Tabelle

Hoofstuk 1 - Inleiding

In die 1990's is 'n versameling Edison-fonograafsilinders, wat aan die Afrikaanse skrywer, digter en staatsman, C.J. Langenhoven, behoort het, deur die Langenhoven Gedenkfonds en Vriende van Arbeitsgenot aan die Universiteit van Stellenbosch geskenk. Meeste van die opnames van Langenhoven en sy naastes was in die eerste en tweede dekade van die 20ste eeu gemaak. Uiteraard is dit 'n onmeetbaar waardevolle kulturele vonds, aangesien daar geen ander oudio-opnames van Langenhoven se stem bestaan nie.

Ongelukkig was meeste van die silinders gebreek of baie ernstig beskadig. Die warm klimaat van Oudtshoorn was ook nie ideale stoortoestande vir die silinders nie. Die silinders kon by langdurige blootstelling aan warm toestande stadig vervorm, aangesien die silinder uit 'n harde tipe was bestaan. Die gebreekte silinders is by die J.S. Gericke-biblioteek deur Mnr. Henri Wirth gerestoureer. Daar is tans ongeveer 10 heel of volledig gerestoureerde silinders.

Aangesien nie slegs die klank op die silinder 'n kultuur historiese erfenis is nie, maar die silinders self ook, moet dit so delikaat moontlik hanteer word. Weens hierdie rede is die Elektries/Elektroniese Departement van die Universiteit van Stellenbosch gevra

Figuur 1 - Die Edison-fonograaf.

om 'n kontaklose, nie-destruktiewe opneemmetode te ontwerp om die klank van die silinders te onttrek. Die befondsing vir die projek is verskaf deur die Langenhoven Gedenkfonds en Vriende van Arbeitsgenot.

'n Foto van 'n Edison-fonograaf word in Figuur 1 getoon. Die Edison-fonograaf is in 1877 deur Thomas A. Edison ontwerp en was die heel eerste oudio-opneemtoestel ter wêreld. Die model van die toestel wat in Langenhoven se besit was, het bestaan uit die kenmerkende fonograafhoring met stilus, 'n rotasiespil, opwenslinger met dryfveer en leiskroef. Twee tipes stilusse is gebruik: een vir opname en 'n ander tydens terugspeel. Tydens opname word 'n skoon wassilinder op die spil gemonteer, die toestel opgewen en in die horing gepraat terwyl die veer afwen. Die spil is teen 'n konstante spoed getransleer deur middel van die leiskroef, om klank in die vorm van 'n heliksgroef in die silinder te graveer. Die klankinligting is in die diepte van die silinderoppervlak gegraveer (vertikale modulasie). Ongeveer twee tot drie minute se klank kan op 'n wassilinder gestoor word.

Die fonograaf se terugspeelstilus oefen meer as 20g druk op die silinderoppervlak uit, dus neem die kwaliteit van die klank drasties af met elke terugspeel van 'n silinder op die oorspronklike fonograaftoestel. 'n Silinder kon slegs 10 maal teruggespeel word voordat die opname heeltemal vernietig is [1].

In Hoofstuk Twee word 'n oorsig van ander wetenskaplikes se pogings gegee om klank op nie-destruktiewe metodes van fonograafsilinders te onttrek. In Hoofstuk Drie word die fisiese afmetings en oudio-eienskappe van die wassilinders bespreek. Hierdie eienskappe gee aanleiding tot belangrike besluite ten opsigte van die ontwerp van die opneemtoestel en monsterfrekwensie. In hierdie tesis is twee nie-destruktiewe metodes van klankonttrekking van Edison-wassilinders ondersoek. Hoofstuk Vier bespreek die eerste klankonttrekkingsmetode, naamlik die mikroskopiese beeldmetode. Klank word vanuit wydtemodulasie sigbaar in mikroskopiese beelde van die silinderoppervlak onttrek. Na gemengde sukses met hierdie metode is 'n meer betroubare metode, wat die dieptemodulasie van die klankgroefie meet, ondersoek. Dit is gedoen deur 'n lasersensor van 'n CD-speler vir dié doel aan te pas. Die ontwerp van die lasersensor-terugvoerstelsel word in Hoofstuk Vyf behandel, tesame met die drie opneemmetodes wat daarmee toegepas is. Die verwerking van die data afkomstig van die dwarsskanderingmetode, word in Hoofstuk Ses behandel. In

Hoofstuk Sewe word die digitale oudiorestourasieprogram en twee spraakverheffings-metodes behandel. Die eerste spraakverheffingsmetode is STSA (short-time spectral attenuation). Dit is die mees gebruikte verheffingsmetode om spraak vanaf ou oudiomedia te verhef. 'n Tweede spraakverheffingsmetode wat ondersoek is, is 'n lineêre voorspellingskoëffisiënt metode. Die evaluasie van die metodes is deur middel van luistertoetse deur 16 luisteraars gedoen. 'n Samevatting, gevolgtrekkings en aanbevelings word in Hoofstuk Agt weergegee.

Hoofstuk 2 - Literatuurstudie

In hierdie hoofstuk word die pogings van ander wetenskaplikes, om klank op innoverende, nie-destruktiewe maniere van Edison-fonograafsilinders te onttrek, vlugtig ondersoek.

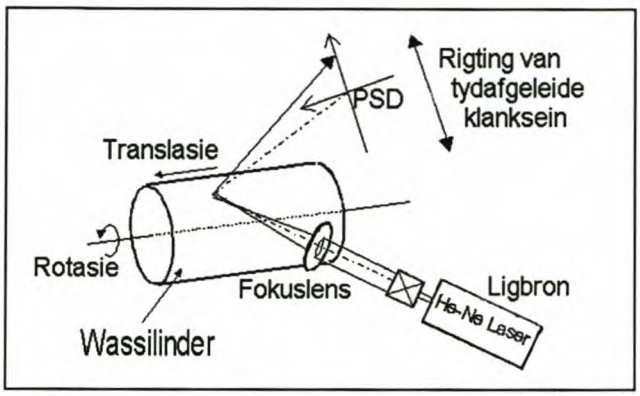
2.1 Verskeie pogings tot klankonttrekking

Verskeie wetenskaplikes en belangstellendes van alle wêrelddele het al menige pogings aangewend om die klank op Edison-wassilinders na moderne oudiomedia oor te dra. Die meeste opneemmetodes maak gebruik van die piësoëlektriese stilus wat in alledaagse vinielplatespelers gebruik word. Hierdeur word die vertikale (diepte) modulasie van die groewe, elektries oorgedra na 'n luidspreker of opneemmedium, byvoorbeeld 'n kassetopnemer of digitaal-na-analoog omsetter in 'n rekenaar, wat die klank versyfer. Van die meer tegnologies gevorderde metodes wat al vir Edison-wassilinder klankherproduksie, aangewend is, word in die res van hierdie hoofstuk behandel.

2.2 Ligtedruk stilus- en laserrefleksieopnames

Asakura en Kawashima [2] van die Hokkaido Universiteit in Japan is gevra om 65 wassilinders met Japanese erfenis na moderne oudiomedia oor te dra. Weens die broosheid van die wassilinders is epoksihars replikas van die oorspronklike silinders gemaak. 'n Negatiewe gietvorm van die silinder is van elastiese viniel-silikonrubber gemaak, aangesien dit baie akkuraat, sag en buigbaar is. Daarna is 'n epoksiharsreplika van die oorspronklike in die gietvorm gegiet.

Ongeveer 40 feitlik onbeskadigde wassilinders is suksesvol opgeneem deur 'n ligtedruk stilus metode. 'n Oppervlakte grofheidmeter met 'n 10μm deursnit stilus is gebruik met 'n stilusdruk van slegs 0.07g op die groefoppervlak. Tydens opname is die silinderrotasie spoed baie stadiger as die oorspronklike opneemspoed – 0.66 tot 0.83opm – weens die 50Hz bandwyde van die sensor. Dit duur ongeveer agt uur om 'n hele silinder op te neem.

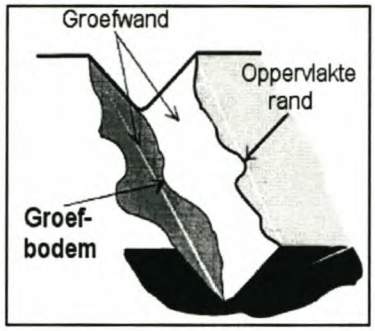


Figuur 2 – Illustrasie om die werking van die laserrefleksiemetode te toon [3].

'n Alternatiewe versnellingsmeter stilus metode is ook vir intydse terugspeel en opname gebruik. 'n Ligtedrukstilus met 250µm deursnee, waarop 'n baie sensitiewe versnellingsmeter gemonteer is, meet die vertikale (diepte) modulasie van die groef. Hierdie sein is die tydafgeleide van die oorspronklike klanksein. Die oorspronklike sein word herwin deur die gemete sein te integreer. Wyeband ruis wat onstaan as gevolg van wrywing tussen die stilus en die wasoppervlak kon verminder word deur die stilusdruk te verhoog, maar dit het die risiko ingehou om die groefoppervlak te verweer.

Die oorblywende 25 silinders was nie geskik om deur 'n stilus teruggespeel te word nie, en 'n kontaklose laserrefleksiemetode is gebruik. Die laserrefleksiemetode werk op die beginsel van geometriese optika. 'n Gaussiese He-Ne (λ = 633nm) laserbundel word op die silinderoppervlak gefokus deur 'n lens in die optiese as. 'n Een-dimensionele posisiesensitiewe sensor (PSD) monster die uitwyking van die gereflekteerde straal se refleksiehoek. Hierdie uitwyking stem ooreen met die tydafgeleide van die groefmodulasie en is dus die klanksein. Die opnames is by die oorspronklike opneemspoed van 150opm gedoen.

Die drie hoof probleme wat oorkom moes word, was die ideale laserkolgrootte, ongewenste eggo's vanaf aangrensende groewe en die groefvolgfout. Die laserkolgrootte bepaal grootliks die kwaliteit van die klank asook die hoeveelheid ruis teenwoordig. Daar is bevind dat 'n groot kolgrootte (162 – 300µm) minder betroubare klank oplewer. Die hoë-frekwensie komponente neem sterk af en sommige medeklinkers is onherkenbaar, omdat die groter kol 'n vergladdingseffek tot gevolg



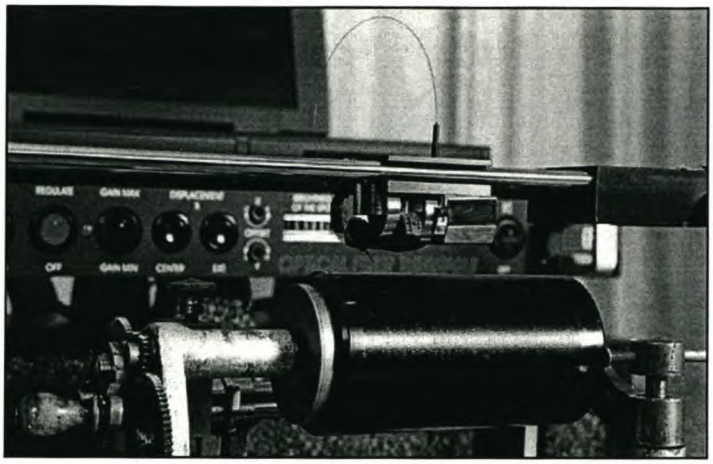
Figuur 3 – Vergrote voorstelling van die groef [4].

het. 'n Kleiner kolgrootte (30 – 130µm) het die mees betroubare klank opgelewer, hoewel 'n te klein kolgrootte weer baie hoë-frekwensie ruis tot gevolg het, as gevolg van spikkelvorming op die PSD deur interferensie van die koherente laserbron.

Ongewenste eggo's is hoorbaar indien die kolgrootte groot genoeg was om die aangrensende groewe te belig. 'n Kolgrootte van 80 – 100µm, wat 'n eggo van 30% van die hoof spraakintensiteit lewer, is as mees gewenste grootte gekies.

'n Groefvolgfout ontstaan wanneer die laserbundel nie presies in die middel van die groef skyn nie, en die bundel sywaarts gereflekteer word as gevolg van die helling van die groefwand. Dit beteken dat die gereflekteerde straal nie die middel van die een-dimensionele PSD direk tref nie, wat 'n daling in intensiteit tot gevolg het. Dit veroorsaak dat die klanksein nie 'n konstante intensiteit handhaaf nie. 'n Twee-dimensionele PSD wat hierdie sywaartse volgfout, tesame met die tydafwyking meet, is in 'n terugvoerstelsel gebruik om vir die volgfout te kompenseer.

Suksesvolle resultate is met die laserrefleksiemetode behaal, hoewel dit steeds nie so betroubaar soos dié van die ligtedrukstilus is nie.



Figuur 4 - Poliak se "optiese draaitafel" [4].

2.3 Die "optiese draaitafel" van Poliak

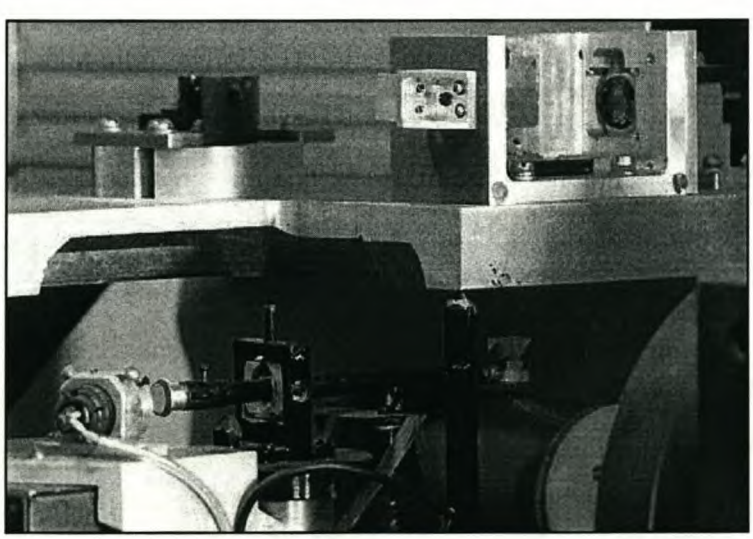
Poliak [4] beweer dat die groefwande die mees betroubare klank bevat, weens twee redes. Eerstens word die groef se oppervlakte uitwyking die meeste deur krappe beskadig. Tweedens gaar onsuiwerhede, soos stof, somtyds in die groefbodem op. Hierdie areas van die groef sal nie sulke betroubare inligting soos die groefwand bevat nie. Figuur 3 toon 'n vergrote voorstelling van die fonograafgroefie.

'n 120μm optiese veselpunt, wat in die groef rus, is gebruik om laserlig na 'n twee-dimensionele PSD te rig. Die veselpunt is so ontwerp om al die laserlig in die vesel met 'n 60° hoek vanaf die vesel se optiese as te refrakteer. Daar bestaan dus geen lig wat vanaf die silinderoppervlak weerkaats word, wat agtergrond ruis kan veroorsaak nie. Die gerefrakteerde lig word deur 'n lens gefokus, waar die horisontale uitwyking op die PSD ooreenstem met die horisontale uitwyking van die groef. Die horisontale uitwyking word vir groefvolging gebruik. Die vertikale uitwyking op die PSD stem ooreen met die dieptemodulasie (klank) van die silindergroef. Die druk wat die optiese vesel op die silinderoppervlak uitoefen is slegs 0.06g, en kan as nie-beskadigend beskou word. Figuur 4 toon 'n foto van Poliak se "optiese draaitafel".

2.4 Optomeganiese laserinterferensie-metode van V.V. Petrov

Petrov [6] gebruik 'n stilus met minder as 2g druk op die silinder. 'n Prisma is aan die stilus gekoppel en groefdieptemodulasie word gemeet deur laserinterferensie. Die beginsel van die metode berus op die klassieke Michelson-interferometer. 'n Laserbundel wat direk na 'n ligsensor (interferometer met drie half-geleier ligsensors) geskyn word, word deur 'n bundelverdeler na die prisma op die stilus gereflekteer. Die prisma weerkaats die bundel terug na die bundelverdeler, wat hierdie bundel ook na die ligsensor rig. Deur konstruktiewe en destruktiewe interferensie van die twee laserbundels, afkomstig van die koherente ligbron, kan hoë en lae ligintensiteit deur die ligsensor gemonitor word. Hierdie intensiteitswisselinge vind elke agtste golflengte ($\frac{1}{8}$) van die ligbron plaas. Die rooi laser wat gebruik is, het 'n golflengte van $\lambda = 0.64\mu m$. Dit beteken dat 'n $\frac{1}{8} = 0.08\mu m$ dieptemodulasieresolusie gehaal kan word, indien gediskretiseer word op die maksimum en minimum ligintensiteitoorgange. Dus kan vir 'n groefmodulasie van ongeveer 10 – 50µm diep, 125 tot 625 resolusiestappe verkry word, wat in die omgewing van 7 tot 10 bis digitale resolusie is. Die resolusie kan tot 0.01µm verhoog word, indien die analoog intensiteitseine vanaf die drie interferometersensors direk verwerk (deur differensiasie) en dan versyfer word.

Die fisiese eksperimentele opstelling is indrukwekkend. Die opneemtoestel weeg drie ton en rus op pneumatiese en rubberskokabsorbeerders. Hierdie opstelling is weer op



Figuur 5 - V.V. Petrov se opneemtoestel [5].

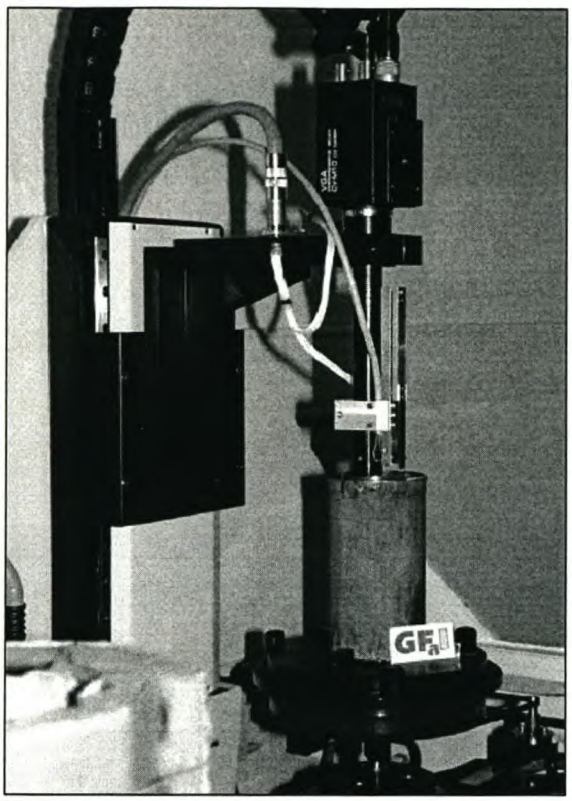
'n 120 ton fondasie gemonteer, wat van die gebou geïsoleer is, om meganiese vibrasies te elimineer. Die silindertranslasiemeganisme loop op 'n lugspoor en die toestel se rotasie-as roteer op luglaers wat 'n minimale vibrasie van 0.01µm op die silinderoppervlak teen 500 rotasies per sekonde veroorsaak. Tydens opname, roteer die opneemtoestel teen tien tot 50 maal stadiger as die oorspronklike opneemspoed van die silinder.

2.5 Direkte opnames vanaf galvaniese silindernegatiewe

Die beeldverwerkingsgroep van die GFaI in Berlyn [1] is gevra om 'n metode te ontwikkel om klank van galvaniese wassilindernegatiewe (galvano's) van die Berlynse Fonograafargief te herwin. Galvano's is negatiewe kopergietsels van oorspronklike wassilinders wat vervaardig is deur die proses van galvanisasie. Dit was gebruik om waskopieë van oorspronklike wassilinders te maak. 'n Galvano se modulasie is in die binneoppervlak van die hol silinder, met die modulasie as riffies in plaas van groefies soos in die wassilinder se geval.

Die opneemtoestel word in Figuur 6 getoon. Dit bestaan uit 'n rotasiestadium en twee translasiestadiums. Die galvano-silinders se aksiale as word vertikaal gemonteer op 'n roterende basis wat deur middel van 'n dryfband deur 'n geratte gs-motor aangedryf word. Die eerste translasiestadium word gebruik vir rifvolging, terwyl die tweede translasiestadium die korrekte druk van die stilus op die galvano se binneoppervlak beheer.

In 'n eerste poging tot klankherwinning, was 'n mediese endoskoop (hoë vergroting kamera) gebruik om die hoogtes van die riffies te fotografeer. Weens 'n tekort aan resolusie, het hierdie metode gemengde sukses opgelewer aangesien slegs galvano's met baie diep modulasie bruikbare klank gelewer het.



Figuur 6 – Galvano-opneemtoestel [1].

'n Ligtedruk (<1g) eliptiese diamant-stilus wat op 'n veer gemonteer is, is as tweede poging ingespan om die rifmodulasie meganies te meet. Die stilus volg dus die hoogteprofiel van die riffie. Aangesien die stilus nie in 'n groefie rus nie, maar op 'n riffie moet loop, was aktiewe stilusbeheer nodig. Hiervoor is die endoskoop in 'n terugvoerstelsel vir akkurate rifvolging gebruik. Die silinders is teen 'n rotasiespoed van 5opm opgeneem.

Goeie resultate, wat vergelykbaar is met die kwaliteit van die oorspronklike terugspeeltoestel, is behaal. Die druk van die stilus was so lig dat mikroskopiese beskouing van die galvano-oppervlak geen skade getoon het nie.

2.6 Gevolgtrekkings

Dit is duidelik dat verskeie gevorderde nie-destruktiewe opneemmetodes reeds beproef is. Meeste van die opneemmetodes het egter direkte kontak met die groefoppervlak deur een of ander vorm van 'n stilus. 'n Streng spesifikasie vir hierdie studie is dat daar egter geen kontak van enige aard met die groefoppervlak mag wees nie. Die laserrefleksiemetode [2] is die enigste ten volle kontaklose nie-destruktiewe metode wat reeds toegepas is. Hierdie metode het wel gunstige resultate gelewer, maar nogtans nie so goed as die ligtedruk stilusmetode nie.

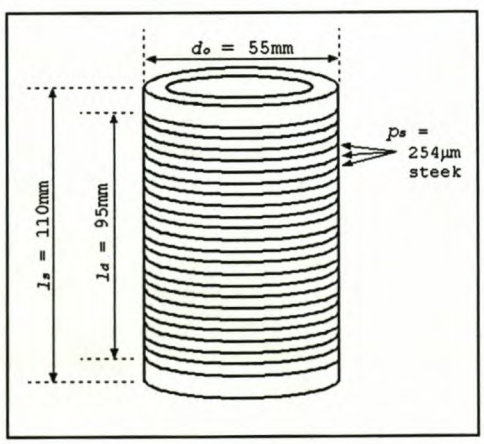
Die begroting is die grootste beperking en 'n koste-effektiewe klankonttrekkings-metode moet ontwerp word met die toerusting wat beskikbaar is.

Hoofstuk 3 - Spesifikasie en ontwerp van die opneemtoestel

In hierdie hoofstuk word die fisiese- en oudio-eienskappe van die Edison-wassilinders weergegee. Vanuit hierdie eienskappe van die silinders kan spesifikasies, soos die benodigde monsterfrekwensie en translasie- en rotasieresolusie, bepaal word. Dit dien as die basis vir die ontwerp van die opneemtoestel.

3.1 Fisiese- en oudio-eienskappe van die Edison-wassilinders

Edison-wassilinders kom in baie verskillende groottes voor. Die Langenhoven-versameling is gelukkig almal eenders. 'n Silinder is $l_s = 110\text{mm}$ in lengte en het 'n buitedeursnit van $d_o = 55\text{mm}$. Die omtrek van 'n silinder is dus $o_o = \pi \times d_o = 172.79\text{mm}$. Die binnedeursnit verander geleidelik in die lengte van die silinder van 43mm na 46mm. Die heliksvormige klankgroefie is met 'n $p_s = 254\mu\text{m}$ steek op die silinderoppervlak gegraveer. Die diepte van die modulasie van hierdie groefie is $10\mu\text{m} < m_d < 50\mu\text{m}$ en verteenwoordig die opgeneemde spraak. Ongeveer $l_d = 95\text{mm}$ van die silinderlengte is gegraveer tydens 'n opname. Die oorspronklike opneemspoed van die silinder was in die omgewing van $T_{oos} = 0.4\text{s}$ per omwenteling of $f_{oos} = 2.5\text{Hz} = 150\text{opm}$. In tyd bevat een silinder ongeveer:



Figuur 7 - Dimensies van silinder.

$$T_{oos} \times \frac{l_d}{p_s} = 0.4 \times \frac{95 \times 10^{-3}}{254 \times 10^{-6}} \approx 150 \text{ sekondes se klank.}$$

Vanuit [2] blyk dit dat die bandwydte van die klank $250 - 4000\text{Hz}$ is, met resonansies by 1kHz en 2.1-2.6kHz.

3.2 Konstruksie en ontwerp van die opneemtoestel

'n Opneemtoestel wat losstaande van die opneemsensor is, is as die beste opstelling vir die opneemtoestel besluit. Dit laat vryheid toe ten opsigte van die opneemsensors wat ondersoek word. Die feit dat die opneemsensor nie deel van die opneemtoestel se struktuur is nie, maak vervangbaarheid van verskillende sensors moontlik.

Die opneemtoestel bestaan uit 'n $10 \times 200 \times 450\text{mm}$ metaalbasisplaat waarop 'n translasieplatvorm en lineêre aktueerder vir translasiebeweging gemonteer is. Die glyvlak van die translasieplatvorm bestaan uit twee silindriese silwerstaalstawe wat parallel aan mekaar op die metaalbasis gemonteer is. Drie teflon kontakpunte van die translasieplatvorm, wat baie lae wrywing glyaksie bewerkstellig, rus op die stawe. Die lineêre aktueerder bestaan uit 'n stappermotor wat aan 'n leiskroef gekoppel word deur 'n ratkassie, en is aan die translasieplatvorm verbind om lineêre beweging moontlik te maak.

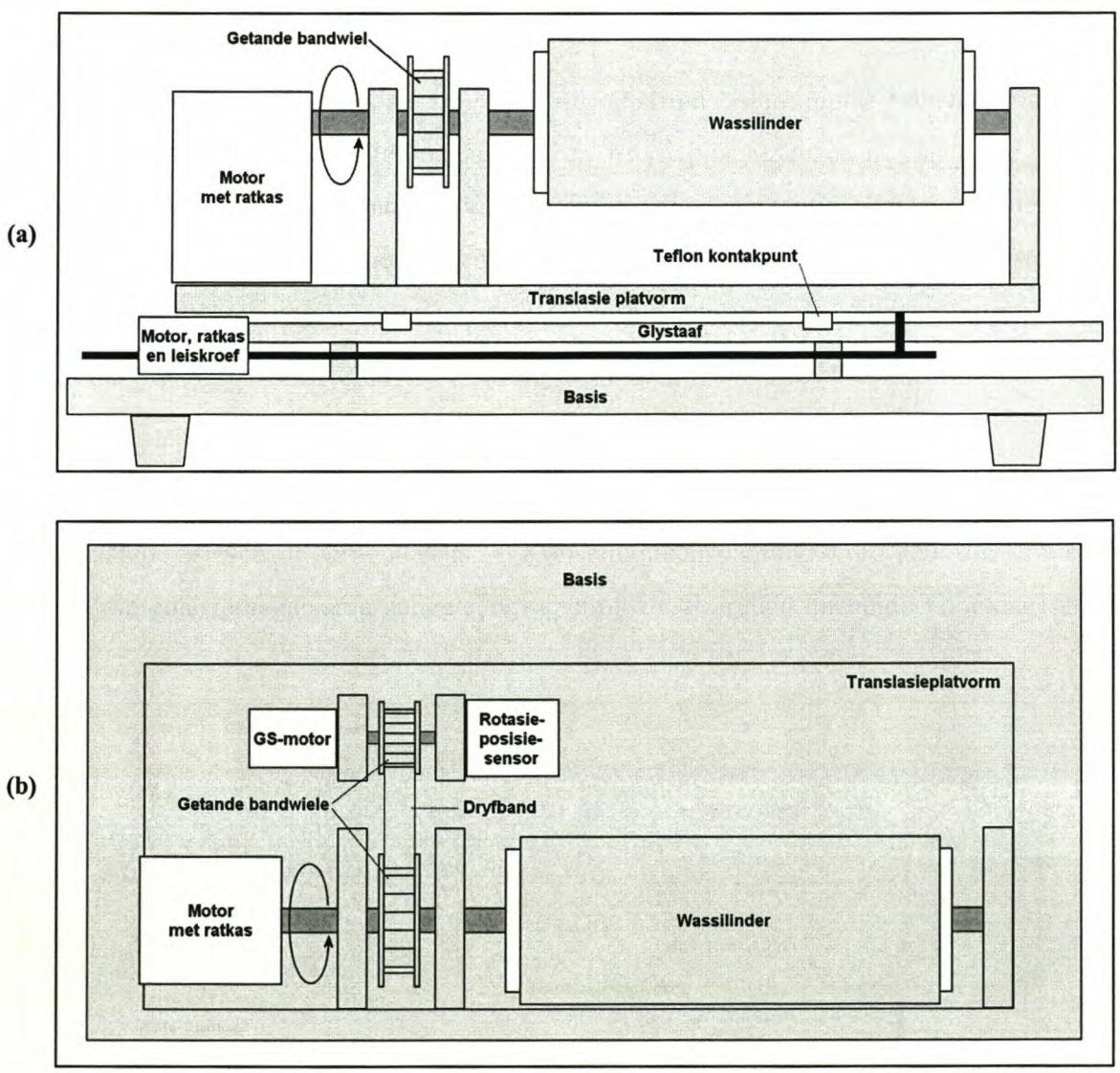
'n Rotasiestappermotor met ratkas, monteerspil, zerospelingmeganisme en

Opsomming van die eienskappe van die wassilinder	
l_s - Silinderlengte (aksiaal)	110 mm
l_d - Geldige datalengte (aksiaal)	95 mm
d_o - Buitedeursnit	55 mm
o_o - Omtrek	172.79 mm
p_s - Steek van heliksgroef	254 µm
m_d - Groef se dieptemodulasie	10 µm < m_d < 50 µm
T_{oos} - Oorspronklike rotasieperiode	0.4 s per omwenteling
f_{oos} - Oorspronklike rotasiefrekwensie	2.5Hz = 150 o.p.m.

Tabel 1 - Opsomming van die eienskappe van 'n tipiese wassilinder.

rotasieposisiesensor is op die $10 \times 140 \times 282$mm aluminium translasieplatvorm gemonteer. Die zerospelingmeganisme en rotasieposisiesensor is vanaf die spil-as deur 'n 2:1 zerospeling bandrat opgerat, om die resolusie van die rotasieposisiesensor te verdubbel. Die rotasieposisiesensor is nie in die huidige eksperimentele opstellings geïmplementeer nie, aangesien die ooplusbeheer van die stappermotors reeds genoegsame monsterfrekwensie tot gevolg het en akkurater geslotelusbeheer onnodig is.

Die zerospelingmeganisme bestaan uit 'n 30W gs-motor wat op dieselfde as as die rotasieposisiesensor gemonteer is. 'n Konstante teenspanning word deur die gs-motor op die rotasiemeganisme aangelê om speling in die ratkas in beide voorwaartse en



Figuur 8 – Die (a) syaansig en (b) bo-aansig van die opneemtoestel (nie volgens skaal nie).

truwaartse aandryfrigtings uit te skakel.

3.2.1 Rotasiemotorkeuse na aanleiding van benodigde monsterfrekwensie

'n Stappermotor met 'n ratkas is vir die rotasie van die silinder te gebruik. Die voordeel van hierdie opstelling is dat dit baie akkuraat 'n groot aantal posisies op die silinderoppervlak diskretiseer vir monstering. Die resolusie wat die stappermotor en ratkas bied, hou direk verband met die frekwensie waarteen die klank gemonster word. 'n Geskikte kombinasie moet dus gevind word vir die geskikte monsterfrekwensie, afgespeel teen wat beskikbaar en finansieel bekostigbaar is.

Volgens die Nyquist-kriterium moet die monsterfrekwensie ten minste meer as twee maal die bandwydte van die analoog sein wees. Met 'n bandwydte van 4kHz vir die klank op die silinder [2] moet die monsterfrekwensie volgens Nyquist ten minste groter as $f_{monster} > 2 \times 4\text{kHz} = 8\text{kHz}$ wees. 'n Standaard monsterfrekwensie wat gebruik word deur persoonlike rekenaars se klankkaarte vir spraakopnames, is 11025Hz, wat hoër as die 8kHz Nyquist-frekwensie is. Dit sal dus gunstig wees om $f_{monster}$ in die orde van 11025Hz of meer te kry vir die silinderopnames. Die aantal monsterposisies per silinderomwenteling om 'n monsterfrekwensie van $f_{monster} = 11025\text{Hz}$ te haal, kan nou bereken word. Met die oorspronklike omwentelingsperiode $T_{oos} = 0.4\text{s}$ is die benodigde monsterposisies per omwenteling:

$$T_{oos} \times 11025 = 0.4 \times 11025 = 4410 \text{ monsters per omwenteling}.$$

'n 1.8°-Stappermotor wat 200 stappe per omwenteling gee, is die stappermotor met die kleinste beskikbare stapinkrement. 'n Geskikte ratverhouding tussen die silinder se rotasie-as en stappermotor kan nou bepaal word:

$$\frac{4410 \text{ monsters per omwenteling}}{200 \text{ stappe per omwenteling}} = 22.05$$

Daar is besluit om 'n 1.8°-stappermotor met 'n 25:1 ratkas van *RS-Components* as rotasie-aandrywing te gebruik. Hierdie kombinasie lewer $200 \times 25 = 5000$ stappe per silinderrotasie. Die klank kan teen 'n effektiewe monsterfrekwensie van:

$$\frac{5000}{T_{oos}} = \frac{5000}{0.4} = 12500\text{Hz}$$ gemonster word, indien die stappermotor met volstappe aangedryf word.

Die fisiese rotasiestaplengte kan as volg bereken word:

$$s_{rot} = \frac{o_o}{\text{stappe/omwenteling}} = \frac{172.79 \times 10^{-3}}{5000} = 34.55\mu\text{m}$$

Om die silinder teen die oorspronklike opneemspoed van 150 opm te roteer moet die stappermotor teen $150 \times 25 = 3750$opm roteer, wat 'n stapfrekwensie van:

$$\frac{3750 \times 200}{60} = 12500 \text{ stappe/s beteken.}$$

Met die unipolêre motoraandrywingskonfigurasie kan hierdie rotasiespoed ongelukkig nie gehaal word nie. 'n Beter konfigurasie vir hoër spoed toepassings is 'n PWM-aandryfkema wat meer draaimoment by hoë rotasiespoed lewer.

3.2.2 Keuse van lineêre aktueerder

Die enigste spesifikasie vir die lineêre aktueerder is dat dit die groefssteek van $p_s = 254\mu\text{m}$ baie akkuraat moet volg. 'n Reeds beskikbare lineêre aktueerder, wat bestaan uit 'n 6°-stappermotor wat deur 'n 10:1 ratkas aan 'n 1mm steek leiskroef gekoppel is, is gebruik. Hierdie lineêre aktueerder kan dus 'n lineêre stapgroote van:

$$s_{tr} = \frac{1\text{mm}}{60 \times 10} = 1.6667\mu\text{m lewer.}$$

Die getal lineêre stappe wat een groefwydte dek (of te wel die getal lineêre stappe wat tydens een rotasie geneem moet word om die groefsteek te volg) is dus:

$$\frac{p_s}{s_{tr}} = \frac{254 \times 10^{-6}}{1.6667 \times 10^{-6}} = 152.4 \text{ stappe per groef.}$$

Teen die oorspronklike opneemspoed van die silinder sal die translasiestapper:

$$\frac{152.4}{T_{oos}} = \frac{152.4}{0.4} = 381 \text{ stappe/s moet neem om die steek van die groef te volg,}$$

wat maklik haalbaar is met die unipolêre stapperaandrywingskema.

Die aantal rotasiestappe wat geneem moet word voordat een translasiestap geneem moet word om die steek van die groef te volg, is:

$$\frac{\text{stappe per rotasie}}{\text{translasiestappe per groef}} = \frac{5000}{152.4} = 32.8084 \text{ rotasiestappe vir 1 translasiestap.}$$

In die volgende hoofstuk word die eerste poging tot klankonttrekking bespreek.

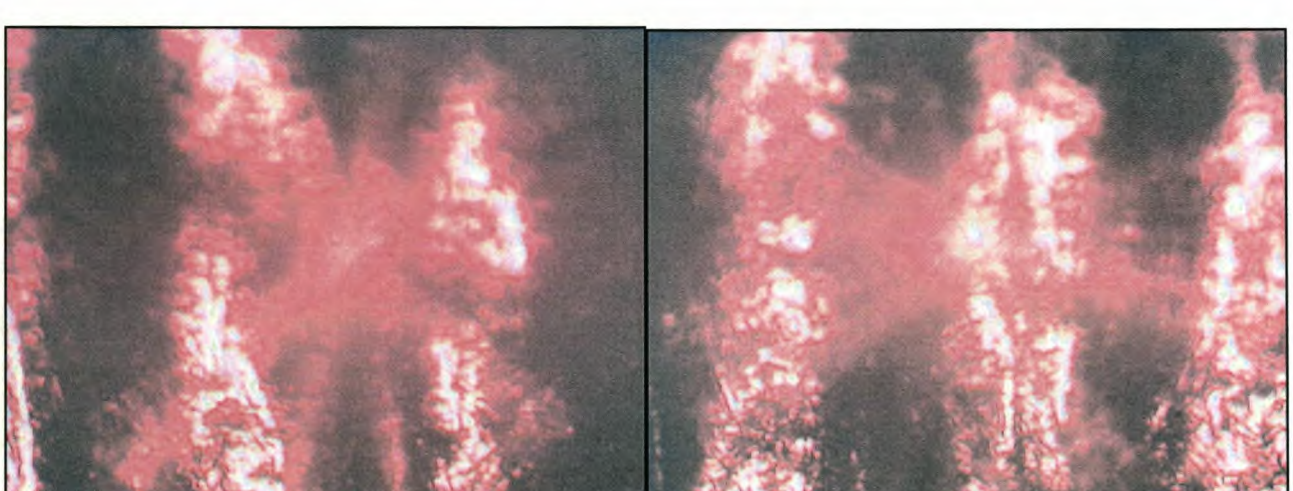
Hoofstuk 4 - Die Mikroskopiese Beeldmetode

In hierdie hoofstuk word die eerste poging tot klankherwinning bespreek. In hierdie hoofstuk sal na die beelde se x- en y-as ook onderskeidelik as die horisontale en vertikale as verwys word.

Die beginsel agter die mikroskopiese beeldmetode is dat daar buiten die groefdieptemodulasie ook groefwydtemodulasie op die oppervlakterand moet wees, aangesien die opneemnaald V-vormig was (sien Figuur 3). Wydtemodulasie is op die silinderoppervlak gevorm, soos die opneemnaald dieptemodulasie in die silinderoppervlak gegrafeer het. Hierdie wydtemodulasie is ook duidelik sigbaar indien die silinderoppervlak deur 'n mikroskoop beskou word.

Die nadeel van hierdie metode is dat die oppervlakte van die silinder die ergste beskadig is deur krapmerke en die inligting nie so betroubaar is nie. Nog 'n nadeel is dat indien die opneemnaald nie presies V-vormig was nie, daar 'n nie-lineêre verband bestaan tussen die wydtemodulasie en die ware dieptemodulasie. Die oorspronklike opneemnaald is egter so erg beskadig dat die stilus vermis is. Dit is dus nie moontlik om die vorm daarvan vas te stel nie.

'n Goedkoop mikroskoop se objektiewe lens, met 'n fokuslengte van 13mm, is deur middel van 'n 110mm buis aan 'n webkamera gekoppel, en gebruik om



Figuur 9 – Twee beelde van twee identiese groewe maar onderskeidelik belig vanaf die linker- en regterkant.

mikroskopiese beelde van die silinderoppervlak te neem. Die wydtemodulasie is duidelik sigbaar vanuit die kontras tussen die silinderoppervlak en die groefwande. Die wydtemodulasie word vanuit die beelde tot klank verwerk.

Die berekening van die benodigde vergroting vir die optiese stelsel word bereken, gevolg deur 'n bespreking van die beligtingsaspekte. Die verwerking van die aaneenskakeling van die opeenvolgende stelle beelde word daarna bespreek, gevolg deur 'n beskrywing van die beeldvasleggingsprogram. Die verwerking van die beelde na klank deur die Viterbi-algoritme word dan bespreek, gevolg deur resultate en gevolgtrekkings.

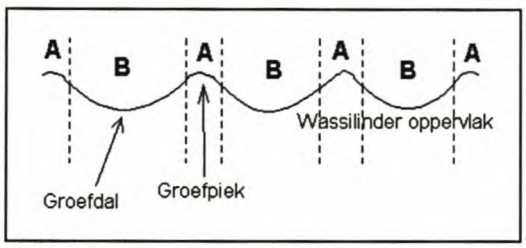
4.1 Benodigde vergroting

Met die 640×480 webkameraresolusie sal dit die voordeligste wees om ten minste 512 pixels oor die groefbreedte (p_s) in te pas vir 9-bis klankversyferingsresolusie. Dit laat ook $640-512=128$ pixels speling vir horisontale groefvariasie as gevolg van vervorming van die silinder. Dit word verlang dat:

$$\frac{p_s}{\text{aantal pixels}}=\frac{254\times10^{-6}}{512}=0.496\mu\text{m}$$ deur 1 pixel verteenwoordig word.

Die effektiewe beeldarea van die "charge coupled device" (CCD) in die webkamera is $4.8\times3.6\text{mm}$. Op die CCD-area verteenwoordig 1 pixel $\frac{4.8}{640}=7.5\mu\text{m}$. Die primêre vergroting van objek tot beeld is dus $\frac{7.5}{0.496}=15.2$.

Weens die horisontale groefvariasies as gevolg van vervorming, is dit wenslik om 'n vergroting ietwat kleiner as 15.2 te kies. Vir die opstelling is die mikroskoop se objektiewe lens, met fokuslengte 13mm, deur 'n 110mm buis aan die webkamera te verbind. Dit beteken die primêre vergroting is $\frac{110}{13}\approx8.5$.



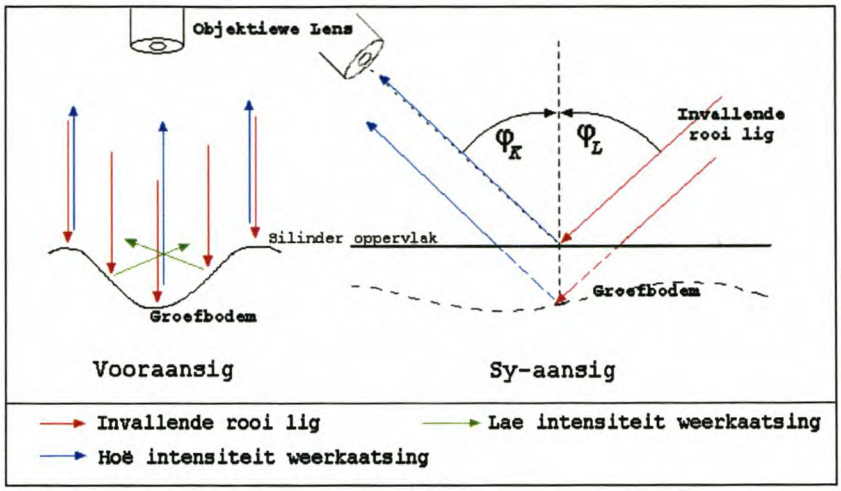
Figuur 10 – Voorstelling van radiale dwarssnit deur drie groewe om hoë kontras-areas te toon. Streek A vertoon helder en streek B donker.

4.2 Beligting van die beelde

'n Sterk kontras word verlang tussen die oppervlak van die silinder en die groefwande, om sodoende die oppervlakterand van die groef sigbaar te maak. Figuur 10 toon helder streke by A en donker streke by B om die verlangde kontras te lewer. 'n Ultra-helder rooi "light-emitting diode" (LED) is gebruik vir die beligting van die groef. Daar is besluit om 'n monochromatiese ligbron te gebruik om enige chromatiese afwyking te beperk. Chromatiese afwyking vind plaas as gevolg van die dispersie van wit lig deur 'n glaslens. Dit beteken dat die verskillende kleure op verskillende posisies op die beeldarea fokus en 'n dowwe beeld tot gevolg het [7].

Die gehalte van die kontras hang ten sterkste af van die beligtingshoek en die kamera se optiese as. Die beste kontras word verkry as die beligtingshoek ten opsigte van die groefrigting (ϕ_L) gelyk is aan die hoek van die kamera se optiese as (ϕ_K), terwyl die kamera en ligbron in dieselfde vertikale vlak parallel aan die groeflengte is.

Vanuit die sketse in Figuur 11 blyk dit duidelik waarom hierdie opstelling die sterkste kontras oplewer. Figuur 11 se vooraansig toon 'n dwarssnit deur die breedte van die groef. Die lig wat loodreg op die silinderoppervlak inval, word by die groefwande weg van die kamera weerkaats (groen pyle), en veroorsaak 'n streek met lae ligintensiteit. Daarteenoor word streke met hoë ligintensiteit veroorsaak by die platter vlakke, soos die silinderoppervlak en groefbodem, omdat die invallende lig na die kamera weerkaats word. Figuur 11 se syaansig toon 'n dwarssnit deur die lengte van die groef. Hier word getoon dat die beste beeld verkry word wanneer die hoek van die



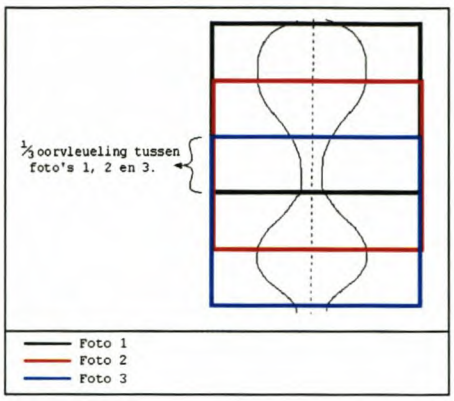
Figuur 11 – Voor- en syaansigdwarssnitte van 'n groef om kontras van silinder oppervlak en bodem teenoor groefwand te toon.

kamera se optiese as gelyk is aan die lig se invalshoek. In die eksperimentele opstelling is die skuinsopname ("*oblique photograph*") opgestel met $\varphi_L = \varphi_K = 45°$.

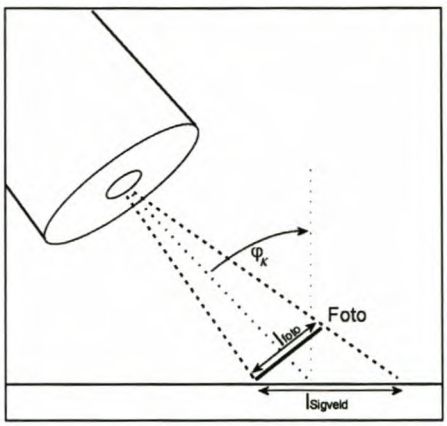
4.3 Berekening van $\frac{2}{3}$ oorvleueling tussen beelde

'n Ongewenste verligte kol kan in die beelde opgemerk word. Dit kan moontlik toegeskryf word aan steurende weerkaatsings vanaf die silinderoppervlak of binne die lensbuis. Om hierdie verskynsel te onderdruk, is besluit om die gemiddelde beeld vanuit drie opeenvolgende oorvleuelende beelde te bereken. Soos die rotasiemotor die groef in die vertikale as van die beeld aanskuif, moet die oorvleuelende derdes van die drie opeenvolgende beelde se gemiddeldes bereken word. Twee opeenvolgende beelde moet dus ten minste $\frac{2}{3}$ oorvleueling hê. Figuur 12 toon die oorvleueling tussen drie opeenvolgende beelde waarvan die gemiddeld bereken word. Met 480 pixels in die vertikale as van die beeld, moet daar dus ten minste $\frac{2}{3} \times 480 = 320$ pixels oorvleueling tussen opeenvolgende beelde wees.

Aangesien die beelde $\varphi_K = 45°$ skuinsopnames is, bestaan daar 'n afstandmetingsfout in die vertikale afstand wat op die beelde vertoon word. Dit word geïllustreer in Figuur 13. Dit veroorsaak 'n resolusieverlies van $\sin(45°) = 0.707$ in die vertikale as van die beeld.



Figuur 12 - Oorvleueling van beelde.



Figuur 13 – Afstandmetingsfout wat bestaan tussen die sigveld en vertikale afstand op die beeld.

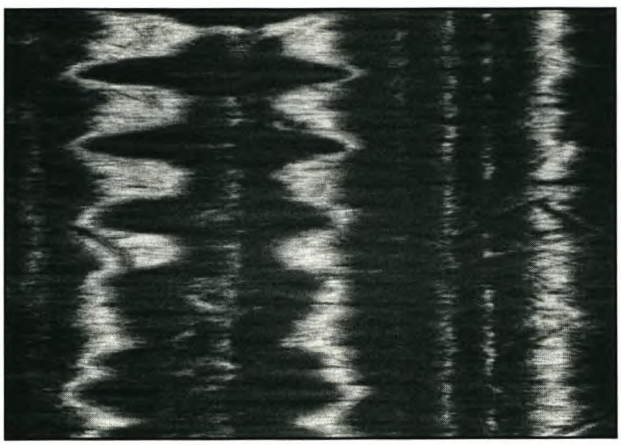
Daar bestaan geen afstandmetingsfout in die horisontale as van die beeld nie, omdat die kamera se optiese as parallel aan die groeflengte gerig is. Vanuit metings op toetsbeelde se horisontale as, is bepaal dat ongeveer 250 pixels die steek van 254µm dek, indien van een groefmiddelpunt tot die naasliggende groefmiddelpunt gemeet word. Die effektiewe horisontale resolusie is dus $\frac{254}{250} = 1.05\mu\text{m/pixel}$. Inaggenome die afstandmetingsfout, kan nou bepaal word hoeveel rotasiestappe per beeld geneem moet word om ten minste $\frac{2}{3}$ oorvleueling tussen opeenvolgende beelde te kry:

$$\frac{\text{pixelskuif tussen foto's} \times \text{fotoresolusie(m)}}{\text{afstandmetingsfoutverhouding} \times s_{rot}} = \frac{160 \times 1.05 \times 10^{-6}}{0.707 \times 34.558 \times 10^{-6}}$$

$$= 6.88 \text{ rotasiestappe/foto}$$

Dus moet ses rotasiestappe tussen opeenvolgende beelde geneem word, wat 'n skuif van 140 pixels tot gevolg het.

Figuur 14 toon 'n seksie van 'n groef waar die $\frac{1}{3}$ gemiddelde beelde in die lengte van die groef aanmekaar gelas is om 'n meer volledige prentjie van die groef te skep. Die modulasie in die groewe is baie duidelik sigbaar. Intermodulasie tussen die aangrensende groewe is ook sigbaar. Die ongewenste verligte kol is ook feitlik uitgeskakel.



Figuur 14 – 'n Gelasde derde-gemiddeldes beeld. Twee aangrensende groewe se intermodulasie is duidelik sigbaar.

4.4 Effektiewe monsterfrekwensie

Met 'n horisontale beeldresolusie van 1.05μm/pixel, kan die effektiewe monsterfrekwensie van die klank bereken word deur die afstandmetingsfout ook in berekening te bring:

$$\frac{\text{afstandmetingsfoutverhouding} \times s_{rot} \times \text{stappe per omwenteling}}{\text{fotoresolusie(m/pixels)} \times T_{oos}}$$

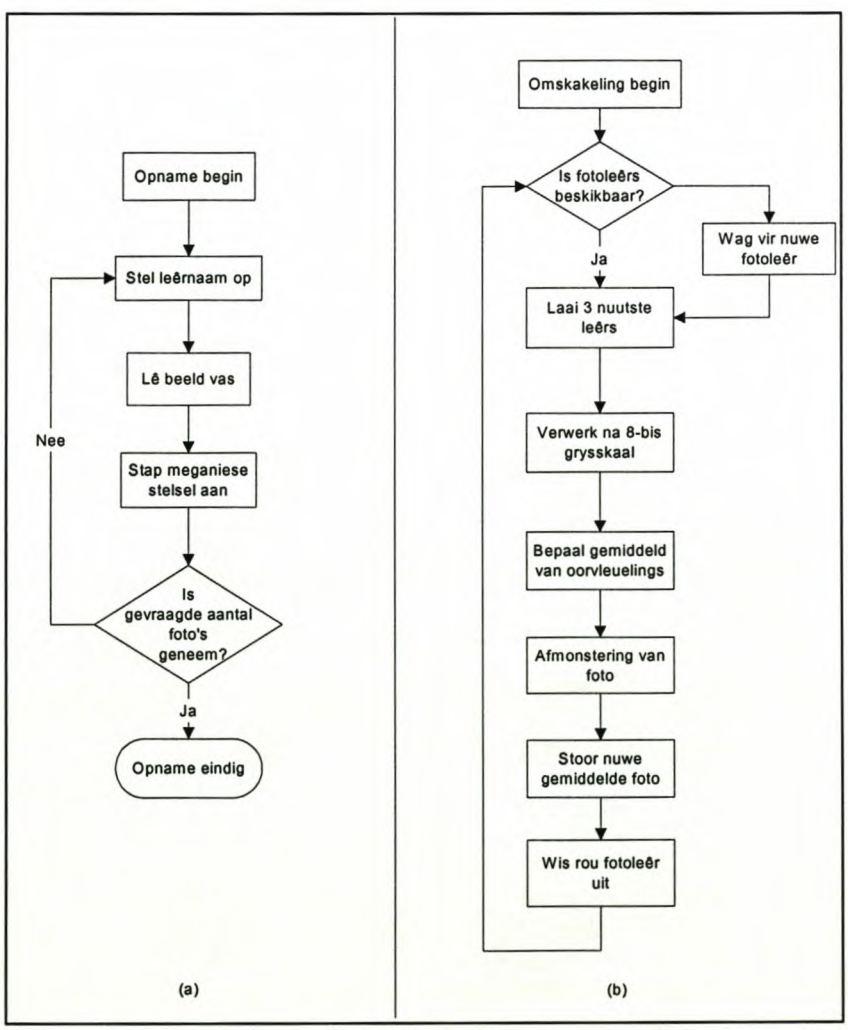
$$= \frac{0.707 \times 34.558 \times 10^{-6} \times 5000}{1.05 \times 10^{-6} \times 0.4} \approx 291\,\text{kHz}$$

4.5 Beskrywing van die beeldvasleggingsprogram

Die program hanteer beeldvaslegging asook die ooplus stappermotorbeheer vir die opneemtoestel. Die vloeidiagram in Figuur 15 toon die werking van die beeldvasleggingsprogram. Beelde word inkrementeel gestoor as 24-bis bisbeeldlêers. Die silinder roteer $\frac{1}{3}$ van die vertikale sigveld (ses rotasiestappe) tussen beeldvaslegging totdat die gevraagde aantal beelde verkry is.

'n Tweede program wat parallel met die beeldvasleggingsprogram loop, word gebruik om die 24-bis RGB-beelde na 8-bis grysskaal beelde om te skakel. Dit is sinvol aangesien die beelde slegs rooi lig bevat en slegs in die intensiteit belanggestel word. Die gemiddeld van die oorvleueling van drie opeenvolgende beelde word dan bereken en afgemonster en die resultaat gestoor as 'n nuwe beeldlêer. Die rede vir hierdie omskakelingsprogram is om kleiner, meer ekonomiese beeldlêers vanuit die rou beelde te skep.

Aangesien die bandwyde van die klank op die silinder 4 kHz is [2], is die effektiewe monsterfrekwensie van 291 kHz, m.a.w. 36 keer oormonster. Stoorspasie kan dus bespaar word deur die beeld met óf 'n faktor drie óf vyf af te monster. Die afmonstering is gedoen deur 'n aantal horisontale pixellyne op die beeld te kombineer



Figuur 15 – (a) toon die vloeidiagram van die beeldvasleggingsprogram. (b) toon die vloeidiagram van die omskakelingsprogram.

tot 'n enkele horisontale pixellyn deur die gemiddeld van die aantal pixellyne te neem. Afmonstering met 'n faktor drie of vyf verlaag die effektiewe monsterfrekwensie na onderskeidelik $\frac{291000}{3} = 97\text{kHz}$ of $\frac{291000}{5} = 58.2\text{kHz}$.

Die spasie is 'n faktor drie kleiner deur die 24-bis beeld na 8-bis om te skakel. Nog 'n faktor drie word bespaar deur slegs die oorvleuelende derde van drie opeenvolgende beelde te stoor. Die faktor spasie wat bespaar word deur afmonstering is óf drie óf vyf, afhangend van die afmonsteringsfaktor. Dit lei tot 'n algehele spasiebesparingsfaktor van óf 27 óf 45 maal.

Met ses rotasiestappe per beeld, dek $\frac{5000}{6} = 833.33$ beelde een rotasie van die silinder. Met ongeveer 370 rotasies vir een silinder kan die nodigde hardeskyfspasie om die hele silinder op te neem, bereken word:

$$\frac{834 \times 370 \times \text{grootte van rou foto (grepe)}}{\text{besparingsfaktor}} = \frac{834 \times 370 \times 921654}{27} \approx 10\text{Gb}$$

OF

$$\frac{834 \times 370 \times \text{grootte van rou foto (grepe)}}{\text{besparingsfaktor}} = \frac{834 \times 370 \times 921654}{45} \approx 6.3\text{Gb}$$

4.6 Klankonttrekking vanuit die beelde

Die klanksein is verberg in die wydtemodulasie van die groewe, wat in die kontras op die beelde gesien kan word. 'n Algoritme wat die kontoer by die skeiding tussen ligte en donker streke vind word verlang. Die Viterbi-algoritme wat die mees waarskynlike toestandsekwensie deur 'n aantal diskrete tydstappe vind, is gebruik om die mees waarskynlike groefwydtekontoer te volg. Voordat die implementering van die algoritme op die beelde bespreek word, word die werking van die algoritme kortliks bespreek.

4.7 Die Viterbi-algoritme

Die Viterbi-algoritme is 'n rekursiewe "optimale" oplossing in 'n mees waarskynlike toestandsekwensie opsig. Dit word gebruik vir die afskatting van die mees waarskynlike toestandsekwensie van 'n diskrete tyd, eindige toestand Markov-proses.

Die Markov-proses word as volg gedefinieer: Tyd k is diskreet. Die toestand i_k by tyd k kan een van 'n eindige aantal M toestande wees, dus is die toestandsruimte $\{1 \le i \le M\}$.

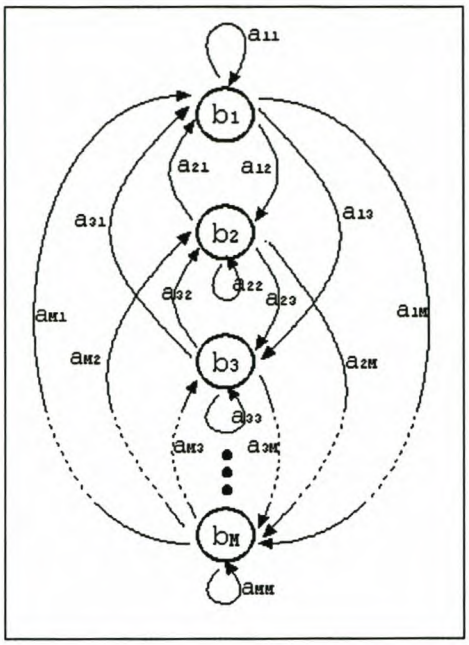
Na aanleiding van oorgangswaarskynlikhede a_{ij} kan die Markov-proses tydens 'n tydstap van sy huidige toestand i na 'n volgende toestand j verander of in sy huidige toestand bly. Die i en j indekse van die oorgangswaarskynlikhede dui onderskeidelik die huidige toestand en volgende toestand tydens 'n tydstap aan. 'n Oorgang a_{12} dui dus die waarskynlikheid aan dat daar van toestand een na toestand twee verander kan word tydens 'n tydstap. Aangesien met waarskynlikhede gewerk word, moet al die uitgaande oorgangswaarskynlikhede van 'n toestand sommeer na een, $\therefore \sum_{j=1}^{M} a_{ij} = 1$.

Die oorgangswaarskynlikhede vir elke toestand kan tydvariant wees, maar is in die meeste praktiese gevalle tydinvariant en dus konstant oor die hele Markov-proses.

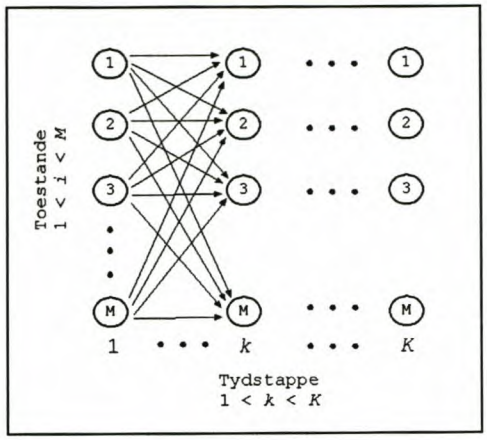
Elke toestand besit ook sy eie koste $b_i(k)$ om in die spesifieke toestand te verkeer. Anders gestel: wat is die kans om in die spesifieke toestand te verkeer op die gegewe tydstap. Hierdie toestandskoste is tydvariant, en speel dus in meeste gevalle die grootste rol in die toestandsekwensieveranderinge.

Figuur 16 toon 'n toestandsdiagram van die Markov-proses met $1 \le i \le M$ toestande met toestandskoste b_i, waarvan al die toestande met mekaar verbind is deur $M \times M$ oorgangswaarskynlikhede a_{ij}. Indien tyd van 0 tot K stap en die begintoestand i_0 en eindtoestand i_K bekend is, bestaan daar 'n mees waarskynlike toestandsekwensievektor $\mathbf{i} = (i_0, \cdots, i_K)$.

Aanvangswaardes om die Markov-proses mee te begin, moet geskep word. Die aanvangswaarskynlikheid vir elke toestand by tyd $k = 1$, word gedefinieer as



Figuur 16 - Toestandsdiagram



Figuur 17 - Trellisdiagram

$\delta_1(i) = \pi_i b_i(1)$ vir $1 \le i \le M$, waar π_i 'n gekose aanvangswaarskynlikheid vir elke toestand is, en $b_i(1)$ die toestandskoste. Indien elke toestand 'n gelyke kans staan om die begintoestand te wees, sal π_i vir elke toestand gelyke gewigte dra. 'n Matriks $\Psi_k(i)$ wat die mees waarskynlike oorgange stoor, word geïnisialiseer na $\Psi_1(i) = 0$.

Nou volg die rekursiewe stappe. Vir $2 \le k \le K$ en $1 \le j \le M$ word die volgende uitgevoer:

$$\delta_k(j) = \max_{1 \le i \le M}[\delta_{k-1}(i)a_{ij}]b_j(k)$$

$$\Psi_k(j) = \arg\max_{1 \le i \le M}[\delta_{k-1}(i)a_{ij}]$$

Die eerste vergelyking toon dat die mees waarskynlike toestandswisseling ($\max[\delta_{k-1}(i)a_{ij}]$) by tyd k met die toestandskoste vermenigvuldig word om die totale gemaksimeerde uitkoms tot op hede vir die Markov-proses in die spesifieke toestand op te lewer. In die tweede vergelyking word die mees waarskynlike oorgange van toestand i na j bloot in 'n matriks gestoor om tydens die terugstapfase van die algoritme die mees waarskynlike toestandsekwensie te vind. Die trellisdiagram in Figuur 17 toon 'n voorstelling van die Viterbi-algoritme wat oor K tydstappe

uitgevoer word. Elke pyltjie wat een toestand met 'n volgende verbind, stel die oorgangswaarskynlikheid a_{ij} voor.

Terminering van die algoritme vind plaas tydens die laaste tydstap K waar die volgende uitgevoer word:

$$P^* = \max_{1 \le i \le M} [\delta_K(i)]$$

$$i_K^* = \arg\max_{1 \le i \le M} [\delta_K(i)]$$

P^* is die maksimum uitkoms van die Markov-proses en i_K^* is die toestand wat die maksimum uitkoms lewer. Die mees waarskynlike toestandsekwensie word gevind deur deur die Ψ-matriks terug te stap:

$$i_k^* = \Psi_{k+1}(i_{k+1}^*)$$

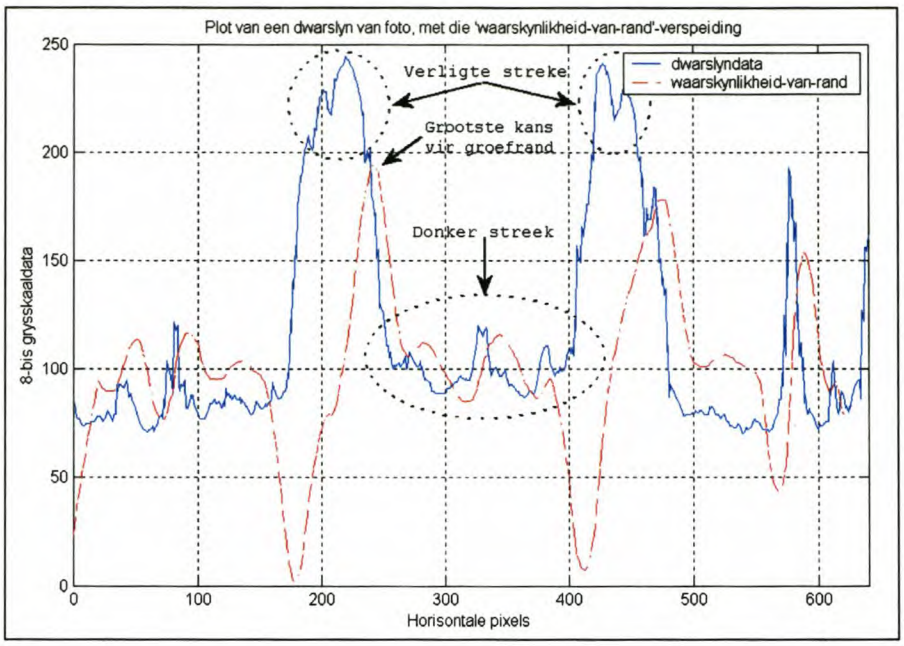
Dit is nie noodwendig nodig om die mees waarskynlike toestandsekwensie vanaf die maksimum uitkoms P^* te bepaal nie. Enige toestand i kan as eindtoestand gekies word en die mees waarskynlike toestandsekwensie vir hierdie eindtoestand kan deur die terugstapfase van die algoritme bepaal word.

4.8 Implementering van die Viterbi-algoritme

Vir die implementering van die algoritme dien die beeld se horisiontale pixels, as die stel toestande i en die vertikale as as die aantal tydstappe k.

'n Toestandskostefunksie $b_i(k)$ moet vanuit die beelddata geskep word. Hierdie kostefunksie moet die waarskynlikheid verteenwoordig dat 'n groefrand by die spesifieke horisontale posisie op die beeld voorkom.

Figuur 18 toon 'n horisontale snit van beelddata, wat voortaan 'n dwarslyn genoem sal word. Die grysskaalwaardes toon duidelik die verligte streke (silinderoppervlak) as hoë datawaardes en donker streke (groefwande) as laer datawaardes. Hierdie verandering van hoë na lae datawaardes kan dus gebruik word om die kontoer van die groefrand te volg. Die tweede plot op Figuur 18 toon die waarskynlikheidsverspreiding vir die dwarslyn. Hierdie verspreiding word gegenereer deur die dwarslyndatapunte te filter met die volgende oordragsfunksie:



Figuur 18 - Plot van dwarslyn, met 'waarskynlikheid-van-rand'-verspreiding.

$$H(z) = \frac{1}{N}\sum_{i=0}^{N-1}(-z^{-i} + z^{-i-N}) \quad \text{met } N = 30\,.$$

Vir 'n skerp negatiewe helling in die dwarslyndata gee die gefilterde weergawe 'n hoë piek, terwyl 'n skerp positiewe helling 'n dal oplewer. Hierdie filter kan ook gesien word as 'n vergladde afgeleide van die dwarslyndata. Hierdie data dien as die toestandskostefunksie $b_i(k)$. In Figuur 18 is die 'waarskynlikheid-van-rand'-plot met 'n afset van 100 geplot om die twee stippe tot dieselfde vlak te kry vir gemaklike besigtiging.

Die oorgangswaarskynlikhede a_{ij} word gekies om te dien as 'n beperking of koste op die grootte van die toestandwisseling. Dit plaas 'n beperking op hoe drasties die toestande (wat deur pixels verteenwoordig word) mag verander. Indien 'n krapmerk of kraak op die beeld voorkom, sal hierdie koste wat in a_{ij} vervat is, sorg dat die toetstandsverandering nie te drasties van die moontlike groefrand afwyk nie. Die oorgangswaarskynlikhede a_{ij} is gekies as 'n verhewe kosinusfunksie met die oorsprong van die kosinus by toestand i en nulle by die oorgange wat wyer as die verhewe kosinus se nulpunte is. Hierdie klokvormige verspreiding laat toe dat die toestande wat die naaste aan die huidige toestand is, hoër prioriteit besit om as die volgende mees waarskynlike toestand gekies te word. Dit veroorsaak dat die kontoer

of toestandsekwensie 'n sekere gladheid het na aanleiding van hoe wyd die kosinusfunksie strek.

Die oorgangswaarskynlikhede met 'n beperking van ses by 'n tydstap k vir 'n spesifieke toestand i lyk soos volg:

$$a_{ij} = \{0\ldots, \underset{a_{i,i-5}}{0}, \underset{a_{i,i-4}}{0.0318}, \underset{a_{i,i-3}}{0.115}, \underset{a_{i,i-2}}{0.218}, \underset{a_{i,i-1}}{0.3}, \underset{a_{ii}}{0.333}, \underset{a_{i,i+1}}{0.3}, \underset{a_{i,i+2}}{0.218}, \underset{a_{i,i+3}}{0.115}, \underset{a_{i,i+4}}{0.0318}, \underset{a_{i,i+5}}{0}, \ldots 0\}$$

Die oorgang met die hoogste waarskynlikheid word altyd deur a_{ii} verteenwoordig. In hierdie voorbeeld sal die huidige toestand nie wyer as vier toestande (pixels) in die volgende tydstap kan spring nie aangesien $a_{ij} = 0$ vir $j \geq i+5$ of $j \leq i-5$.

Die Viterbi-algoritme is apart op die linker- en regtergroefrande uitgevoer. Hiervoor is die beelddata vanaf die groefmiddel in 'n linker- en regtersnit verdeel om die getal toestande (horisontale pixels) te verminder en om moontlike toestandsekwensiespronge tussen die linker- en regtergroefrand te voorkom. Die verwerkingstyd van die Viterbi-algoritme is kwadraties afhanklik van die getal toestande, aangesien M oorgange vir M toestande bereken moet word. Die getal toestande wat vir die algoritme gevoer word speel dus die grootste rol in verwerkingstyd. Die verwerkingstyd is slegs lineêr afhanklik van die getal tydstappe wat die algoritme moet uitvoer.

Die klank word verkry deur die wydtemodulasie vanuit die linker- en regter-groefrandkontoere te bepaal. Dit word gedoen deur die verskil van die twee kontoere by die ooreenstemmende tydstappe te neem.

4.9 Resultate van die Viterbi-algoritme

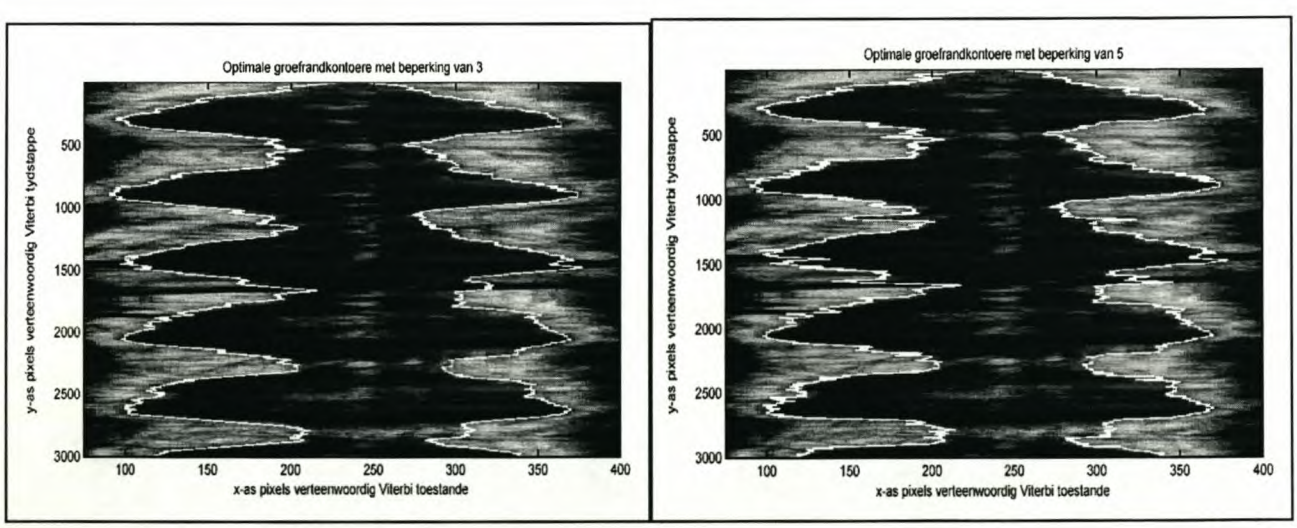
Bewerking is met Matlab onder die Linux bedryfstelsel op 'n *Pentuim™III* 1.5GHz verwerker met 512Mb geheue uitgevoer. 'n Beelddatasnit met 300 toestande (horisontale pixels) en 3000 tydstappe (vertikale pixels), het elk ongeveer 8 minute se verwerkingstyd geneem om die kontoere op te lewer. 'n Kleiner beelddatasnit van 200×3000 pixels het $3\frac{3}{4}$ minute geneem om te verwerk. Dit korreleer met die teorie

dat die verwerkingstyd kwadraties toeneem na aanleiding van die getal toestande ($(\frac{300}{200})^2 = 2.25$ keer). Die 3000 tydstapmonsters verteenwoordig

$$\frac{\text{aantal monsters}}{f_{monster}} = \frac{3000}{58.2 \times 10^3} = 51.5\text{ms se klank.}$$

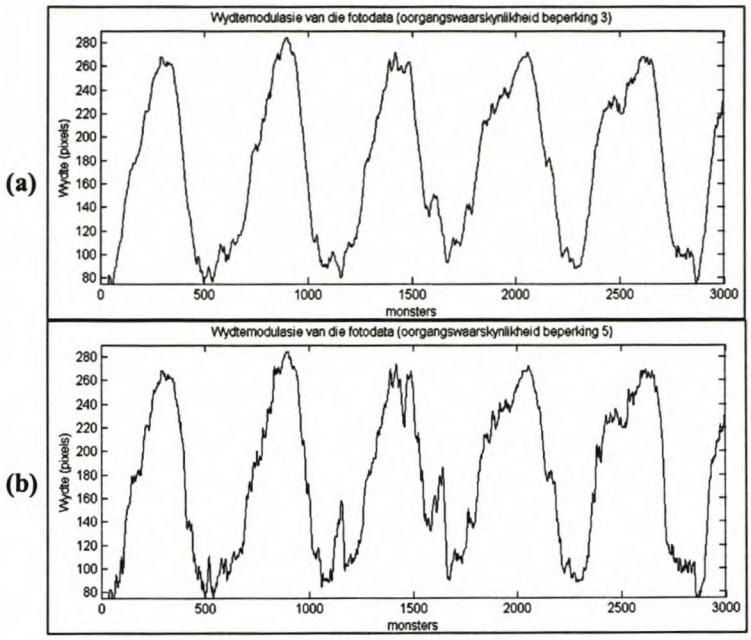
Figuur 19(a) toon die linker- en regterkontoere met 'n beperking van drie op die oorgangswaarskynlikheidsverspreiding. Dit beteken dat daar van een tydstap na die volgende, nie meer as 'n maksimum van een pixel horisontale uitwyking mag wees nie. Figuur 19(b) toon die linker- en regterkontoere met 'n beperking van vyf op die oorgangswaarskynlikheidsverspreiding. Dit beteken dat daar van een tydstap na die volgende, nie meer as 'n maksimum van drie pixels horisontale uitwyking mag wees nie.

Figuur 20(a) toon die wydtemodulasie wat verkry is vanuit die verskil tussen die linker- en regterkontoere. Die beperking op die oorgangswaarskynlikheids-verspreiding is drie. Figuur 20(b) toon die wydtemodulasie vir dieselfde klanksegment met 'n beperking van vyf op die oorgangswaarskynlikheidsverspreiding. Figuur 20(a) en Figuur 20(b) vertoon minder hoë-frekwensie ruis en stap beter oor krapmerke in vergelyking met die resultaat in Figuur 20(b). Krapmerke is sigbaar



(a) **(b)**

Figuur 19 - In (a) word die linker- en regtergroefrandkontoere met 'n beperking van drie op die oorgangswaarskynlikheidsverspreiding getoon terwyl, (b) die linker- en regtergroefrandkontoere met 'n beperking van vyf op die oorgangswaarskynlikheidsverspreiding toon vir dieselfde seksie van beelddata.



Figuur 20 - Snit van klank verkry deur die groefwydtemodulasie.

in die figure by tydmonsters 1460, 1650 en 1900. Weens die strenger beperking van drie, sukkel die algoritme soms om skielike skerp groefkurwes te volg en word hierdie kurwes deur 'n reguit lyn benader, soos opgemerk kan word in Figuur 20(a) van monsters 400 tot 500 asook 900 tot 1000.

4.10 Gevolgtrekkings

Hoewel resultate vir die mikroskopiese beeldmetode behaal is, is gevoel dat hierdie resultate en data nie die mees betroubare tot beskikking is nie. Dit is toe te skryf aan 'n paar redes. Die groefrand is ongelukkig die onbetroubaarste plek om data van die silinders te onttrek, aangesien die silinderoppervlak die meeste deur kepe en krapmerke verwring word.

Die groefrand is ook nie die ware modulasie van die klanksein nie, aangesien die klank in die diepte van die silinder gegraveer is. Indien die oorspronklike opneemnaald nie presies V-vormig was nie, bestaan daar 'n nie-lineêre verband tussen die groefwydtemodulasie en die ware dieptemodulasie. Na verwagting sal daar dus meer betroubare inligting in die dieptemodulasie van die groefie wees, aangesien dit die ware klanksein verteenwoordig, en ook minder deur 'n oppervlaktekrapmerk beïnvloed word.

Intermodulasie tussen groewe veroorsaak ook steurende periodiese eggo's. Die intermodulasie blyk duidelik vanuit die wydtemodulasie, soos een groef met wye modulasie in die naasliggende groef inkerf. Hierdie eggo-effek sal vanselfsprekend nie teenwoordig wees in die dieptemodulasie nie.

Die kwantiseringsresolusie van die klank vir hierdie metode is ongelukkig ook laag. Die maksimum resolusie behaal vanuit die resultate in Figuur 20 is ongeveer 200 kwantiseringsvlakke. Dit verteenwoordig met ander woorde klankkwaliteit van tussen sewe- en agt-bisresolusie. Die verwerking van die beelde na klank neem ook baie lank, en die stoorspasie wat die rou data opneem is baie groot.

Weens hierdie redes, is besluit om 'n tweede opneemmetode te ondersoek, wat data vanuit die hele groefprofiel benut. In die volgende hoofstuk word die lasersensor-terugvoerstelselmetode bespreek.

Hoofstuk 5 - Die lasersensor-terugvoerstelselmetode

'n Lasersensor-terugvoerstelselmetode is ontwikkel om die dieptemodulasie van die klankgroefie te meet. Weens 'n beperkte begroting kon 'n kommersiële hoë akkuraatheid laserverplasingsmeter nie bekostig word nie. Wat wel tot beskikking was, was 'n lasersensor van 'n CD-speler. Hierdie lasersensor is aangepas om die funksie van 'n mikrodieptemeter te verrig. 'n Terugvoerstelsel is ontwerp om die onegalige, nie-silindriese vervorming van die wassilinder te volg. Die steurende 2.5Hz-ossillasie wat deur hierdie onegaligheid veroorsaak word, word sodoende uitgeskakel.

Drie opneemmetodes word ondersoek: voorwaartse en truwaartse opneemmetode en dwarsskandering. Tydens opname word die gemete sein vanaf die lasersensor versyfer deur 'n analoog-na-digitaal-omsetter en op 'n hardeskyf van 'n rekenaar gestoor. Hierdie digitale data word dan verder tot klank verwerk.

In hierdie hoofstuk word agtergrond oor 'n CD-speler se lasersensor eerstens bespreek. Die aanpassings wat aan die lasersensor aangebring is om as 'n mikrodieptemeter te funksioneer, word daarna bespreek. Die detailontwerp van die aangepaste lasersensor se terugvoerstelsel word bespreek, gevolg deur die voorwaartse opneemmetode, truwaartse opneemmetode en dwarsskandering. Laastens word eenvoudige seinverwerking wat op die data toegepas is, bespreek.

5.1 Interne komponente en werking van 'n CD-speler se lasersensor

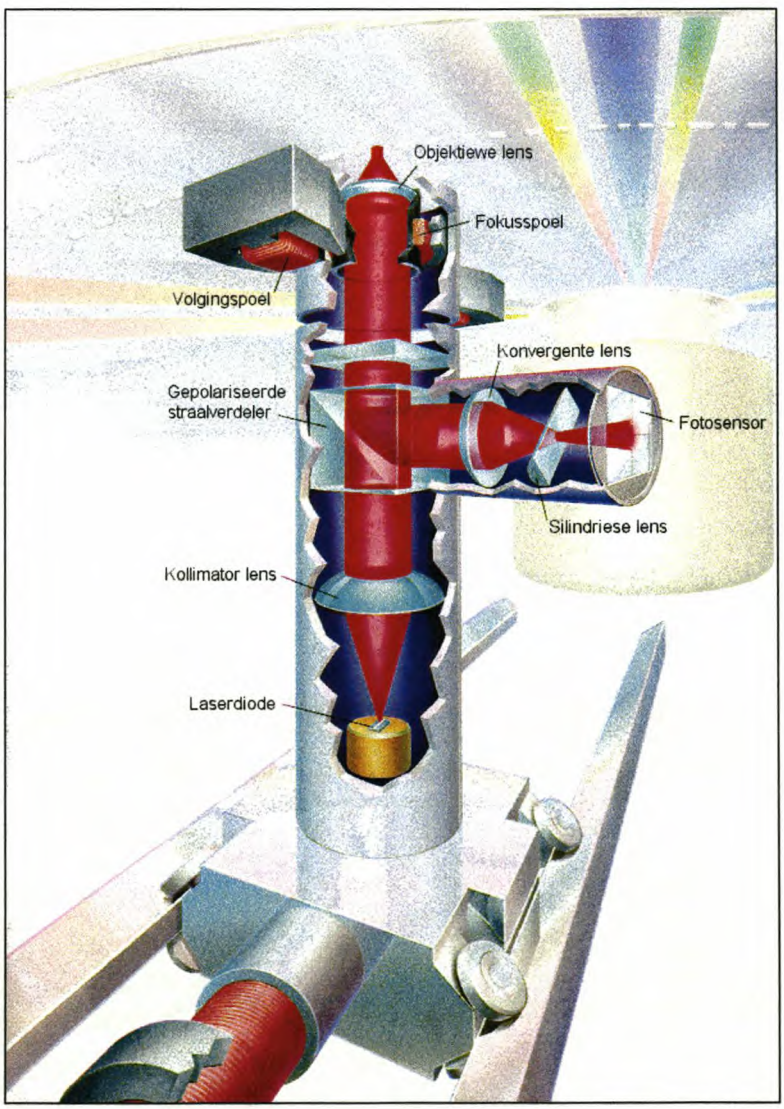
'n CD-speler se lasersensor funksioneer by fisiese dimensies in die orde van mikro- tot nanometers. Hierdie eienskap van die lasersensor se fokusmeganisme, maak dit moontlik om die lasersensor aan te pas om as 'n mikrodieptemeter te funksioneer.

'n Laserskyf se digitale inligting is in 'n groefie, wat van die middel van die polikarbonaatskyf na buite spiraal, geëts. Die steek van die spiraal is 1.6μm [8], [9], wat beteken dat die deursnit van die laserbundel se kolgrootte by die fokuspunt dus ook in hierdie orde is. Dit beteken dat die lasersensor geskik is vir implementering op

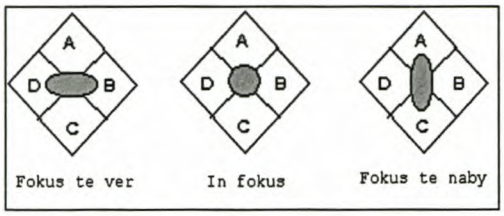
die wassilinders, aangesien die kolgrootte heelwat kleiner is as die silinder se groefbreedte (p_s).

Hier volg 'n kort oorsig van die komponente en werking van 'n tipiese CD-speler se lasersensor met verwysing na Figuur 21:

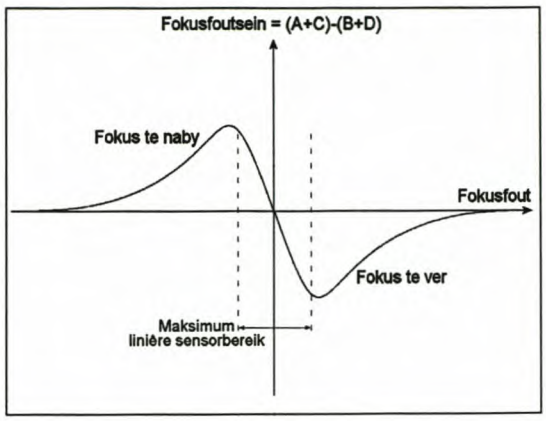
Die laserbundel van die sensor word gegenereer deur 'n GaAs of ander halfgeleier laserdiode. Die laserdiode in 'n alledaagse CD-speler straal 'n 1mW infrarooibundel met golflengte $\lambda = 780\text{nm}$ by die objektiewe lens van die sensor uit [9], [10]. Die laserbundel gaan vanaf die laserdiode deur 'n diffraksierooster, wat die laserbundel in 'n hoofbundel met 'n reeks sybundels verdeel. Die hoofbundel en twee naasliggende sybundels word deur die lasersensor gebruik vir die aflees van die data en volging van



Figuur 21 - Interne komponente van 'n CD-speler se lasersensor [8].



Figuur 22 - Fokus van laserbundel op fotosensors [8].



Figuur 23 - Fokusfoutsein van lasersensor.

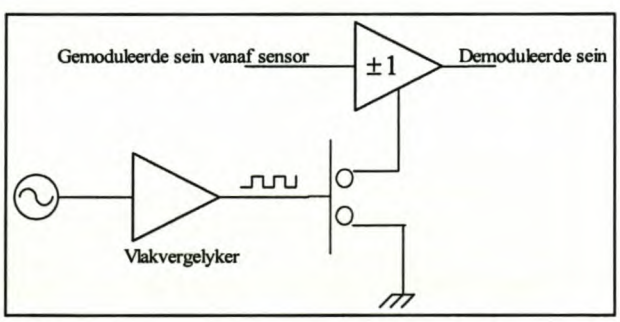
die spiraalgroefie. Vanaf die diffraksierooster gaan die laserbundel deur 'n kollimatorlens, vanwaar dit deur 'n gepolariseerde bundelverdeler en kwartgolflengtevertrager gaan. Die laserbundel is nou sirkulêr gepolariseer. Nadat die bundel vanaf die laserskyfoppervlak weerkaats het, is die inkomende bundel nou loodreg ten opsigte van die uitgaande bundel gepolariseer. Dit veroorsaak dat die inkomende bundel nie deur die gepolariseerde bundelverdeler deurgelaat word nie, maar na vier fotodiodes weerkaats. Voordat die bundel die fotodiodes bereik, gaan dit deur 'n silindriese lens. Die doel van die silindriese lens is om die astigmatiese fokus wat in Figuur 22 geïllustreer word, te bewerkstellig. Die vier fotodiodes is as vier kwadrante (A, B, C en D) van 'n vierkant gerangskik. Wanneer die laser te ver uit fokus is, word kwadrante B en D meer verlig as A en C, en wanneer die laser te naby uit fokus is, word kwadrante A en C meer verlig as B en D. Al vier fotodiodes sal ewe veel verlig word indien die laser in fokus is. Vanuit die verskil in stroommetings van diodes (A+C) en (B+D) word 'n fokusfoutsein verkry om die laserbundelfokus op die laserskyfoppervlak te beheer. Die oordragsfunksie van die fokusfout word in Figuur 23 getoon. Hierdie verandering in fokusafstand, wat direk verband hou met die afstand van die reflekterende oppervlak, is gebruik om die dieptemodulasie van die wassilinder te meet.

5.2 Aanpassings aan die lasersensor om as mikrodieptemeter te funksioneer

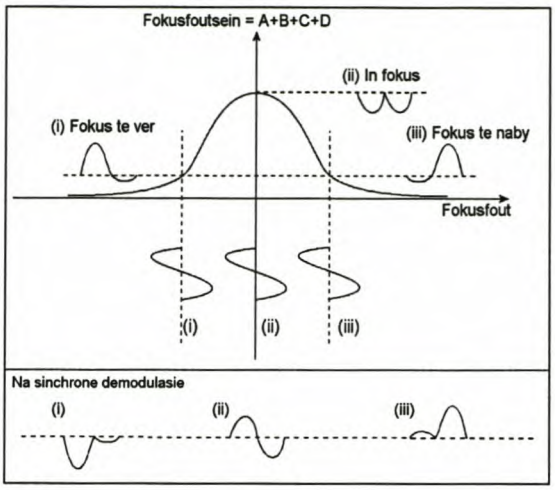
Weens die onreëlmatige silinderoppervlak, wat drasties verskil van die gladde, blink, reflekterende oppervlak van 'n CD, was die fokusfoutsein nie na wense om as 'n dieptemeting gebruik te word nie. Na deeglike ondersoek is bepaal dat 'n sommasie van die vier fotodiodeseine $(A+B+C+D)$ 'n meer robuuste sein lewer, wat minder sensitief is vir die onreëlmatige silinderoppervlak. Wanneer die laser nou in fokus is, word 'n maksimum seinwaarde vanaf die fotodiodes verkry. Figuur 25 toon die oordragfunksie van die aangepaste lasersensor. Om die fokusafwyking te bepaal, is 'n wisselsein op die fokusspoel aangelê. Deur die wisselsein vanaf die fotodiodes sinchroon te demoduleer, word 'n positiewe of negatiewe fokusfoutsein verkry wanneer die laser te ver of te naby uit fokus is, soos getoon in Figuur 23. 'n Terugvoerstelsel is ontwerp om die verandering in groefdiepte te monster na aanleiding van die verandering in die fokusfoutsein. Die detailontwerp en implementering van die terugvoerstelsel word in die volgende afdelings, na die werking van die sinchrone demodulator, bespreek.

5.3 Werking van die sinchrone demodulator

Figuur 24 toon 'n diagrammatiese voorstelling van die sinchrone demodulator. Bylae C bevat 'n meer volledige skematiese stroombaandiagram. Die demodulator bestaan uit 'n vlakvergelyker, analoog skakelaar en demodulasieversterker. Die



Figuur 24 - Diagrammatiese voorstelling van die sinchrone demodulator.



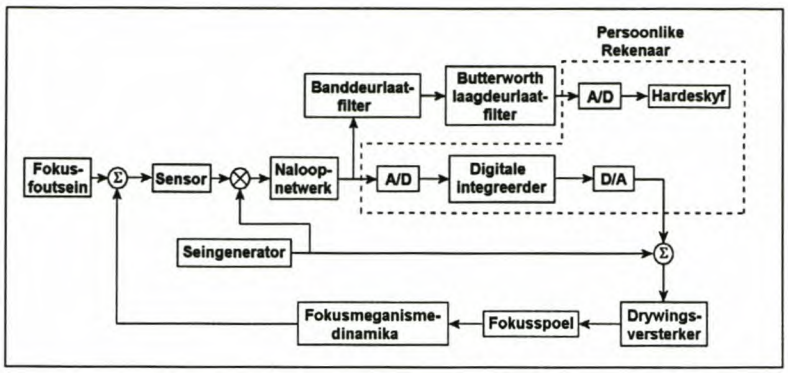
Figuur 25 - Oordragsfunksie van aangepaste lasersensor.

vlakvergelyker is 'n OV (operasionele versterker) met die positiewe intree verbind aan die wisselsein en die negatiewe intree wat die drempelvlak verstel deur 'n potensiometer verbind tussen die ±10V toevoerspannings. Die vlakvergelyker se uittree is aan die analoogskakelaar verbind, wat veroorsaak dat dit aanskakel wanneer die modulasiesein bo die drempelvlak is en afskakel onder die drempelvlak. Hierdie skakeling veroorsaak dat die demodulasieversterker, waarvan die positiewe intree aan die skakelaar verbind is, skakel tussen 'n omkerende en nie-omkerende versterker met 'n aanwins van ±1. Gevolglik word 'n gemiddeld positiewe, negatiewe of zerosein by die uittree van die demodulator verkry indien die laserbundel te naby, te ver of in fokus is ten opsigte van die silinderoppervlak, soos getoon in Figuur 25.

5.4 Ontwerp van die lasersensor se terugvoerstelsel

Die silinder is weens vervorming nie meer perfek sirkelvormig silindries nie. Hierdie vervorming veroorsaak 'n steurende $f_{oos} = 2.5\text{Hz}$ ossillasie wanneer die silinder teen die oorspronklike opneemspoed geroteer word. Aangesien integraalbeheer zeroposisievolgfout verseker, is dit geskik om die onegalige rotasie van die silinder te volg en sodoende die effek van die 2.5Hz ossillasie uit te skakel.

Figuur 26 toon 'n blokdiagram van die terugvoerstelsel wat geïmplementeer is om die lasersensor as mikrodieptemeter te gebruik. Die fokusspoel van die sensor is op 'n veer gemonteer. Deur meting van die impulsweergawe van die sensor, is bepaal dat



Figuur 26 - Blokdiagram van aangepaste lasersensor-terugvoerstelsel.

die dempingsfaktor 0.05 is, met 'n natuurlike frekwensie van 33Hz (207.35 rad/s). Dit plaas 'n beperking op die frekwensie van die wisselsein wat op die fokusspoel aangelê moet word, aangesien die poolpaar by 33Hz 'n 20dB per dekade daling in die frekwensieweergawe van die fokusspoel tot gevolg het. Met ander woorde, hoe hoër die wisselsein se frekwensie gekies word, hoe minder is sy effek op die fokusspoel. Die wisselseinfrekwensie is gekies as 50Hz. 'n Krohn-Hite filter model 3550 4de-orde Butterworth-laagdeurlaatfilter is gebruik om 40dB verlies in die 50Hz wisselseinfrekwensie te verkry, na sinchrone demodulasie. Dit plaas 'n beperking op die bandwydte van die laserterugvoerstelsel aangesien die afsnyfrekwensie van die laagdeurlaatfilter $\frac{50}{3} = 16.667 \approx 17\text{Hz}\,(106.8\,\text{rad/s})$ is. Die bandwydte van die klank op die silinder is $250 - 4000\text{Hz}$ [2]. Die rotasiespoed van die silinder moet met 'n faktor $\frac{4000\times3}{50} = 240$ afgeskaleer word om die klank binne die bandwydte van die 4de-orde Butterworth-laagdeurlaatfilter te laat val. Die bandwydte van die klank is nou dus $\frac{250\text{Hz}}{240} = 1.04\text{Hz}\,(6.53\,\text{rad/s})$ tot 17Hz. Die omwentelingsperiode om aan die terugvoerstelsel se bandwydtespesifikasie te voldoen is dus

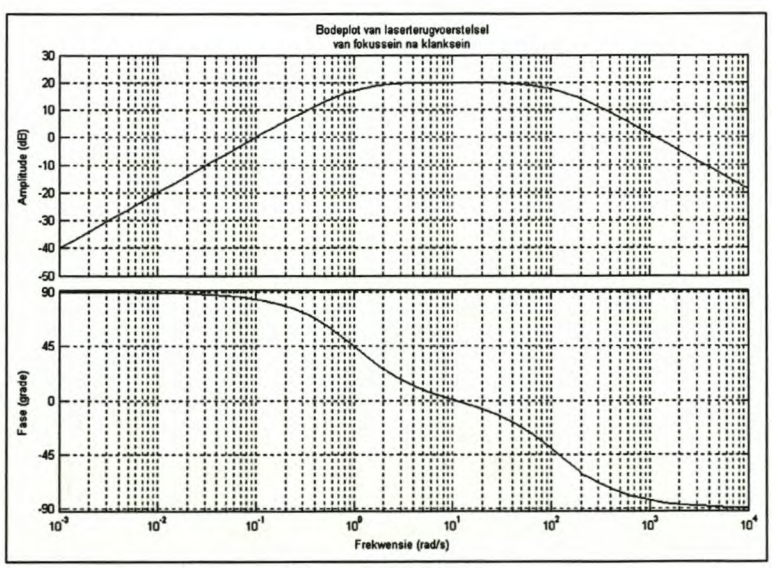
$$T_{op} = T_{oos} \times 240 = 0.4 \times 240 = 96\text{s}.$$

Die integreerder is digitaal geïmplementeer. Die rede hiervoor is om die fokus van die laser te herwin indien 'n groot kraak of vermiste deel van die silinder onder die sensor deur beweeg. Wanneer die sensor dus heeltemal uit fokus raak, word die stappermotors gestop, die beheerlus verbreek en deur die fokusbereik van die sensor geveeg totdat die sensor se fokusfoutsein weer nul is.

'n Naloopnetwerk met 'n pool by 18.55Hz (116.55 rad/s) is na die gedemoduleerde fokusfoutsein geplaas om aanwins en bandbeperking aan die terugvoerstelsel te gee. Hierdie aanvullende bandbeperking voorkom dat steurende eksterne vibrasies hoër as 18.55Hz 'n ernstige invloed op die stelsel sal hê.

Die aanwins van die lasersensor word bepaal deur die aanwinste van die drywingsversterker, fotosensor, sensorseinversterker en sinchrone demodulator. Die aanwins van hierdie komponente is deur meting tot eenheid verstel, binne die 1.04 – 17Hz bandwydte wat op die stelsel van toepassing is. Die aanwins van die sommeerder, naloopnetwerk en digitale integreerder word nou verstel om 'n gunstige geslotelus weergawe vir die stelsel te gee. Na aanpassing van die aanwinste in die stelsel, is die frekwensieweergawe in Figuur 27 behaal, soos geëvalueer in Matlab. Die aanwinste van die sommeerder, naloopnetwerk en digitale integreerder is onderskeidelik $K_{Som} = \frac{1}{3}$, $K_{NN} = 10$, en $K_I = \frac{1}{3}$.

Figuur 27 toon 'n aanwins van 20dB vir die gevraagde bandwydte met laag- en hoogafsnyfrekwensies van onderskeidelik 0.16Hz (1rad/s) en 19Hz (119rad/s) vir die geslotelusoordragsfunksie vanaf die fokusfoutsein tot by die klanksein. Die 20dB/dekade hoogafsnyhelling sal eksterne vibrasies voldoende onderdruk.



Figuur 27 - Bodeplot van die geslotelus oordragfunksie van die fokusfoutsein na klanksein.

5.5 Implementering van die lasersensor se terugvoerstelsel

Die skematiese stroombaandiagram van die lasersensorterugvoerstelsel, soos dit met analoogelektronika geïmplementeer is, word in bylae C getoon. Die 50Hz gemoduleerde sensorsein gaan deur 'n sensorseinversterker en word daarna deur die sinchrone demodulator gedemoduleer. Die gedemoduleerde sein dien as die fokusfoutsein met 'n oordragsfunksie wat in Figuur 23 getoon word. Die koppelkapasitor tussen die sensorseinversterker en sinchrone demodulator dien as 'n hoogdeurlaatfilter met laagafsnyfrekwensie by:

$$\omega_L = \frac{1}{RC} = \frac{1}{22\times10^3\times330\times10^{-9}} = 137.7 \text{ rad/s } (22.9\text{Hz}).$$

Enige gs-afset wat op die sensorsein mag voorkom, word deur die koppelkapasitor verwerp, aangesien 'n gs-afset die demodulasieproses minder sensitief sal maak vir dinamiese fokusfoutafwykings.

Die naloopnetwerk is geïmplementeer as 'n eerste orde laagdeurlaatfilter met 'n verstelbare aanwins. Hiervoor is 'n OV met 'n 10kΩ potensiometer as intreeweerstand gebruik, met die RC-netwerk as terugvoer. Die RC-netwerk lewer 'n hoogafsnyfrewkensie van $\omega_H = \frac{1}{RC} = \frac{1}{39\times10^3\times220\times10^{-9}} = 116.55 \text{ rad/s } (18.55\text{Hz})$. Die aanwins van die filter is verstel tot $K_{NN} = 10$.

Die digitale integreerder is geïmplementeer op 'n persoonlike rekenaar toegerus met 'n ASA3.0 A/D-D/A ISA-kaart ontwerp deur die Universiteit van Stellenbosch se Elektriese/Elektorniese Departement. Die ASA3.0 kaart het vier 12-bis A/D-kanale en twee 12-bis D/A-kanale wat tussen ±10V werk. Een A/D-kanaal en een D/A-kanaal is onderskeidelik as die digitale integreerder se intree- en uitreekanale gebruik.

Deur Euler se metode vir die oplossing van differensiaalvergelykings toe te pas, word 'n algebraïese vergelyking verkry wat die integraal benader [11]:

$$u(k) = u(k-1) + K_I T e(k).$$

$u(k)$ en $u(k\text{-}1)$ stel onderskeidelik die huidige en vorige integraalwaarde voor, $e(k)$ die huidige gemonsterde intreewaarde van die integreerder, K_I die integraalaanwins en T die monsterperiode van die integreerder.

'n Monster word na elke rotasiestap geneem. Indien die integraal na elke rotasiestap opgedateer word, is die integraal se monsterfrekwensie:

$$f_s = \frac{1}{T} = \frac{\text{stappe per rotasie}}{T_{op}} = \frac{5000}{96} = 52.08\text{Hz}$$

Die faktor waarmee die huidige monsterwaarde $e(k)$ in die algebraïese vegelyking van die integraal vermenigvuldig moet word, is dus $K_I \times T = \frac{1}{3} \times \frac{1}{52.08} \approx \frac{1}{156}$. Met implementering van die algebraïese vergelyking vir die integraal, word $e(k)$ deur 156 gedeel.

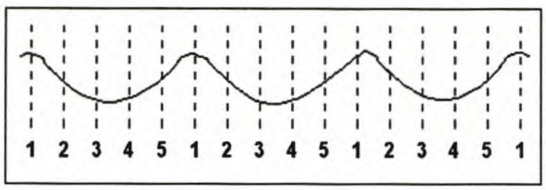
Die integreerder is as 'n analoë komponent in die terugvoerstelsel ontwerp, en daarna digitaal geïmplementeer. Die digitale houbaan dra ekstra tydvertraging en fase by die terugvoerstelsel. Die ekstra tydvertraging is benaderd die helfte van die monsterperiode $\frac{T}{2}$ en die ekstra fasebydra, benaderd $-\frac{\omega T}{2}$ waar ω die 0dB-kruisfrekwensie is [12]. Met die integreerder se aanwins van $K_I = \frac{1}{3}$, is die integreerder se kruisfrekwensie $\omega_{0\text{dB}} = \frac{1}{3}$ rad/s. Die ekstra fase wat deur die houbaan veroorsaak word, is dus benaderd $-\frac{\omega_{0\text{dB}} T}{2} - \frac{1}{2\times3\times52.08} = -0.003$ rad. Hierdie fasebydrae is weglaatbaar klein.

Die uittree van die integreerder, asook die 50Hz wisselsein is aan 'n sommeerder verbind. Die sommeerder het 'n aanwins van $K_{Som} = \frac{1}{3}$ en sy uittree is aan die intree van die drywingsversterker verbind wat die fokusspoel dryf.

Die klanksein word vanaf die naloopnetwerk se uittree afgetap. Dit gaan deur 'n eerste orde banddeurlaatfilter en daarna deur die 4de-orde Butterworth-laagdeurlaatfilter (Krohn-Hite-filter, model 3550) en word dan gemonster deur die tweede A/D-kanaal van die ASA3.0 kaart en op die rekenaar se hardeskyf gestoor.

5.6 Opneemmetodes

Drie verskillende opneemmetodes is toegepas om klank van die silinders te onttrek. Eerstens is bloot saam met die lengte en steek van die groef in die voorwaartse rigting van die klank geskandeer. Met die tweede opneemmetode is truwaarts geroteer en klank is van agter na voor opgeneem. Die rede vir hierdie vreemde opneemprosedure



Figuur 28 – 'n Voorstelling van 'n deursnit deur drie groewe met 'n opnamestel wat bestaan uit vyf individulele opnames elk $\frac{1}{5}$ groefbreedte uitmekaar gespasieer.

is bloot om nog data op 'n alternatiewe manier van die silinder te onttrek met die doel om dit met die voorwaartse data te kombineer. Met 'n paar aanpassings aan die reeds aangepaste lasersensor, is 'n derde opneemmetode, waar dwars oor die groewe skandeer word, ook toegepas.

5.6.1 Voorwaartse opneemmetode

Hierdie opneemmetode is die eenvoudigste en mees vanselfsprekend. Die dieptemodulasie van die groefie word bloot deur die lasersensor gemonster soos die groef onder die lasersensor verby roteer met die groefsteek p_s. Een stel opnames bestaan uit vyf individuele opnames, wat elk 'n vyfde van die groefbreedte ($\frac{1}{5} p_s$) langs mekaar gespasieer is. Figuur 28 toon 'n voorstelling van die vyf opnames, wat een opnamestel vorm oor drie groefbreedtes. Die rede agter die vyf opnames is om data vanuit verskillende posisies in die groefwande te onttrek. Die datastel word so vergroot en die opnames kan gekombineer word om 'n beter sein-tot-ruis-verhouding te verkry.

Hoewel vyf opnames in 'n stel opgeneem word, is slegs vier hiervan prakties bruikbaar. Die rede hiervoor blyk uit Figuur 28. Een van die vyf opnames van die opnamestel gaan uiteraard by die posisie tussen twee groewe val (opname een in Figuur 28). Dit beteken dat hierdie spesifieke opname van baie swak gehalte is, met eggo-effekte deurdat dit inligting van beide aangrensende groewe bevat. Hierdie opname word by verwerking nie in berekening gebring nie.

Opname van die klank geskied deur die opneemprosedure te aktiveer om 'n sekere aantal datapunte te versamel en stoor. Na versameling van die gevraagde aantal datapunte, word die stelsel teruggestap tot by die opname se beginposisie. Daarna

word 'n vyfde van 'n groefbreedte ($\frac{1}{5}p_s$) getransleer en die opneemprosedure word herhaal. Om die hele groefbreedte te dek word hierdie prosedure vyf maal herhaal en so word die vyf individuele opnames van die opnamestel geskep.

5.6.2 Voorwaartse opneemprogram

In Figuur 29(a) word die vloeidiagram van die voorwaartse opneemprogram getoon. 'n Opname begin deur 'n prosedure uit te voer wat die lasersensor op die oppervlak van die silinder fokus. Hier volg die beskrywing van die "Verkry Fokus"-prosedure wat deur die vloeidiagram in Figuur 29(b) getoon word:

- Die maksimum waarde van die integraaluittree, wat die fokusbeheersein is, word op D/A0 uitgesit.

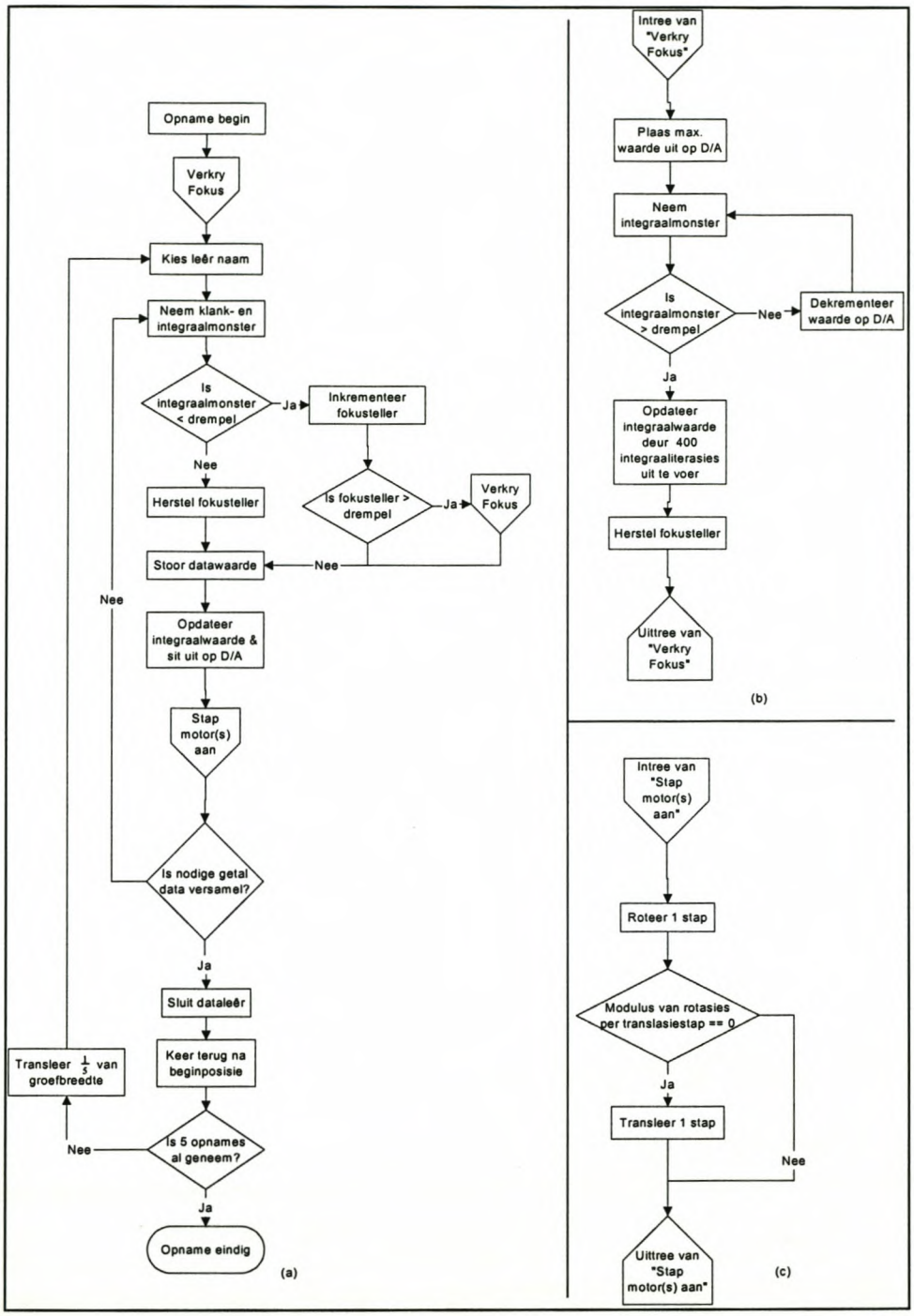
- Die integraalintree, wat die fokusfoutsein is, word gemonster en vergelyk met 'n drempelwaarde om te bepaal of die lasersensor al meer gefokus is.

- Indien die fokusfoutsein steeds kleiner as die drempelwaarde is, word die fokusbeheersein gedekrementeer en die nuwe waarde van die fokusfoutsein gemonster en weer met die drempel vergelyk. Hierdie proses word herhaal totdat die gemonsterde fokusfoutsein groter as die drempelwaarde is.

- Die silinderoppervlak is nou binne die fokusbereik van die lasersensor. Die fokusfoutsein word gevolglik tot bestendigheid gebring deur 'n aantal ingetraaliterasies uit te voer.

- Die fokusteller word herstel.

Die datalêer word opgestel nadat fokus van die lasersensor verkry is. Monsters van onderskeidelik die klank- en fokusfoutsein word geneem. Die sensor kan fokus verloor wanneer 'n silinder met 'n vermiste deel onder die sensor verbyroteer. Hierdie verskynsel word gemonitor deur te toets of die fokusfoutsein groter as 'n sekere drempelwaarde is. Indien die fokusfoutsein kleiner as die drempelwaarde is, word 'n fokusteller geïnkrementeer. Daar word aanvaar dat die lasersensor nie meer op die silinderoppervlak gefokus is nie wanneer 'n groot aantal fokusfoutseinmonsters onder die fokusfoutdrempel val, en die fokusteller sy drempelwaarde oorskry. Die "Verkry Fokus"-subprosedure word geroep om die sensor weer te fokus.



Figuur 29 - Vloeidiagram van voorwaartse opneemprogram. (a) toon die hoofprogram met (b) "Verkry fokus"-subprosedure en (c) "Stap motor(s) aan"-subprosedure.

Indien die fokus wel in orde is, word die fokusteller herstel en die klankseinwaarde gestoor. Die integraalwaarde word opgedateer en op die D/A uitgesit. Die meganiese

stelsel word daarna een stap aangeskuif. Die subprosedure in Figuur 29(c) toon hoe die rotasie- en tanslasiebewegings gesinchroniseer word:

- Die rotasiemotor neem een rotasiestap.
- Na elke 33 rotasiestappe moet een translasiestap geneem word. Dit gebeur wanneer die modulus (of res) van die aantal rotasiestappe gedeel deur die aantal rotasiestappe per translasiestap gelyk aan nul is.

Die monstering, stoor en integraalopdatering word herhaal terwyl die gevraagde aantal datapunte nog nie opgeneem is nie. Die datalêer word gesluit en die meganiese stelsel na die beginposisie van die opname herstel sodra die gevraagde aantal monsters versamel is. Indien al vyf individuele opnames van die opnamestel nog nie opgeneem is nie, transleer die stelsel 'n vyfde van 'n groefbreedte ($\frac{1}{5} p_s$) en word die volgende opname van die opnamestel opgeneem. Die hoofprogram termineer wanneer al vyf opnames van die opnamestel opgeneem is.

5.6.3 Truwaartse opneemmetode

Die truwaartse opneemmetode is soortgelyk aan die voorwaartse opneemmetode, met die groot verskil dat die hele meganiese stelsel in trurat loop. Dit beteken dat die silinder van agter na voor opgeneem word. Die rede onderliggend aan die implementering van hierdie opneemmetode is dat klein kepe, krake en ruis effens anders as in die voorwaartse opneemmetode opgeneem kan word. Die voorwaartse en truwaartse opnamestelle kan nou gekombineer word om 'n beter resultaat te lewer.

Soos met die voorwaartse metode is vyf individuele opnames oor die groefbreedte geneem om 'n truwaartse opnamestel te vorm. Die opneemprogram is ook soortgelyk aan dié wat vir die voorwaartse opnames gebruik is, met die verskil dat die meganiese stelsel slegs truwaarts roteer en transleer.

5.6.4 Resultate van die voorwaartse en truwaartse opneemmetodes

Die sein-tot-ruis-verhouding (SNR) word bereken as die verhouding van die seindrywing of –variansie (σ_s^2) teenoor die ruisdrywing of –variansie (σ_r^2). Die seinvariansie word verkry deur die afskatting van die ruisvariansie van die ruiserige sein se variansie af te trek ($\sigma_s^2 = \sigma_{s+r}^2 - \sigma_r^2$).

Opname	1	2	3	4	Gemiddeld van 4 opnames
SNR individueel (dB)	-0.37	4.61	1.99	0.85	1.77
SNR gemiddeld (dB)	4.35				

Tabel 2 - Vergelyking van SNR tussen individuele opnames van die opnamestel en die gemiddeld van die opnamestel.

Ruis word verkry deur die stilteseksies van die opnames te identifiseer, en hierdie stiltesnitte in 'n vektor te stoor. Die variansie van die ruis word van die ruisvektor afgeskat deur van Matlab se 'var'-funksie gebruik te maak.

Die ruiserige sein se variansie word verkry deur die 'var'-funksie op die ruiserige klanksein toe te pas. Daarna word die SNR deur die volgende vergelyking bereken:

$$\mathrm{SNR}_{\mathrm{dB}} = 10\log\left(\frac{\sigma_{s+r}^2 - \sigma_r^2}{\sigma_r^2}\right).$$

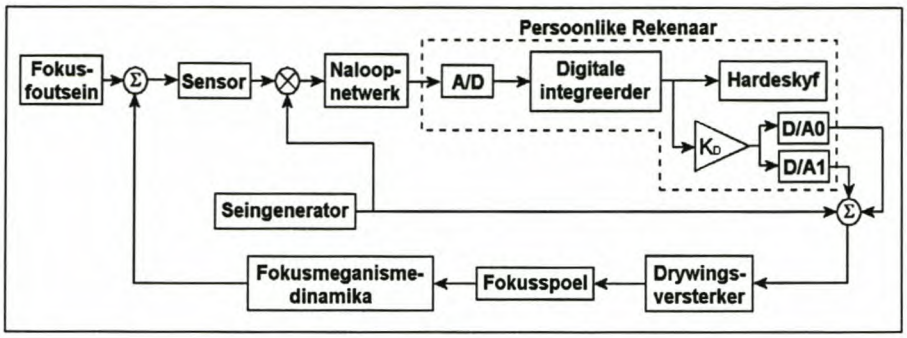
Die sein-tot-ruis-verhouding (SNR) vir die opnames word in Tabel 2 - getoon. Dit is duidelik dat 2.58dB gewen word wanneer die gemiddeld van die vier bruikbare opnames in die opnamestel gebruik word in plaas van slegs 'n enkele opname. Hierdie openbaring het aanleiding gegee tot die dwarsskandering opneemmetode wat vervolgens bespreek gaan word.

5.6.5 Dwarsskandering

'n Derde opneemmetode wat die absolute diepte van die groef meet, is geïmplementeer. Die sensorterugvoerstelsel is aangepas vir hierdie spesifieke toepassing. Figuur 30 toon die blokdiagram van die dwarsskanderingmetode se terugvoerstelsel.

Dwarsskandering skandeer dwars oor 'n aantal groefbreedtes, terwyl die integraal die absolute diepte van die groefprofiel volg. Die idee is om 'n aantal monsters van die groefprofiel tot 'n klankmonster te verenig, om sodoende maksimale inligting van die hele groefprofiel te gebruik om klank te onttrek.

Die sensorseinversterker met sinchrone demodulator is identies aan die opstelling van die voorwaarts- en truwaartse opneemmetodes. Die naloopnetwerk se aanwins is deur



Figuur 30 - Blokdiagram van die dwarsskanderingmetode se terugvoerstelsel.

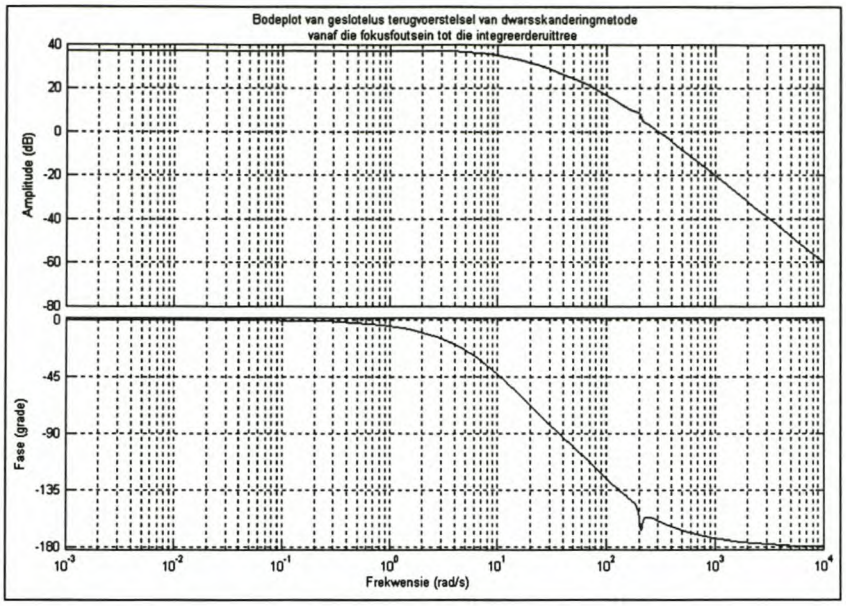
inspeksie verstel sodat die sensorsein se maksimum lineêre bereik presies tussen die ±10V toevoerspanning swaai voordat dit deur die rekenaar gemonster word. Die rede hiervoor is om maksimum kwantisering van die sensorsein tussen die ±10V kwantiseringsbereik van die A/D te verkry om die kwantiseringsfout te verminder. 'n Aanwins van $K_{NN} = 28$ het aan hierdie vereistes voldoen.

Albei 12-bis D/A's op die ASA3.0 kaart is gebruik om die uittreeresolusie van die integraal te verdubbel na 13-bis. Vir hierdie doel is die twee D/A se uittrees gesommeer. 'n Digitale aanwins $K_D = \frac{1}{8}$ is net na die integraal geplaas om die 16-bis integraalwaarde na 13-bis af te deel. Hierdie aanwins dra ook by tot die ontwerp van die terugvoerstelsel om die gevraagde bandwydte te haal.

'n Translasiespoed van 240 stappe per sekonde is gekies. Met 152.4 translasiestappe oor een groefbreedte, beteken dit dat die frekwensie waarteen die groewe onder die sensor deurbeweeg $\frac{240}{152.4} = 1.5748\text{Hz}\,(9.89\,\text{rad/s})$ is. Daar moet dus ontwerp word vir 'n hoogafsnyfrekwensie ten minste hoër as 1.5748Hz vir die geslotelus terugvoerstelsel.

Die sommeerder aanwins K_{Som} en integraal aanwins K_i moet nou nog aangepas word om die gevraagde frekwensieweergawe te lewer. Simulasie in Matlab toon 'n geskikte frekwensieweergawe vir die aanwinste $K_{NN} = 28$, $K_D = \frac{1}{8}$, $K_I = 30$ en $K_{Som} = 0.11$. Figuur 31 toon die geslotelus frekwensieweergawe van die stelsel. Die hoogafsnyfrekwensie is $\omega_c = 12.8\,\text{rad/s}$.

Indien die integraal na elke translasiestap opgedateer word, beteken dit dat die integraal se monsterfrekwensie 240Hz is. Die faktor waarmee die huidige

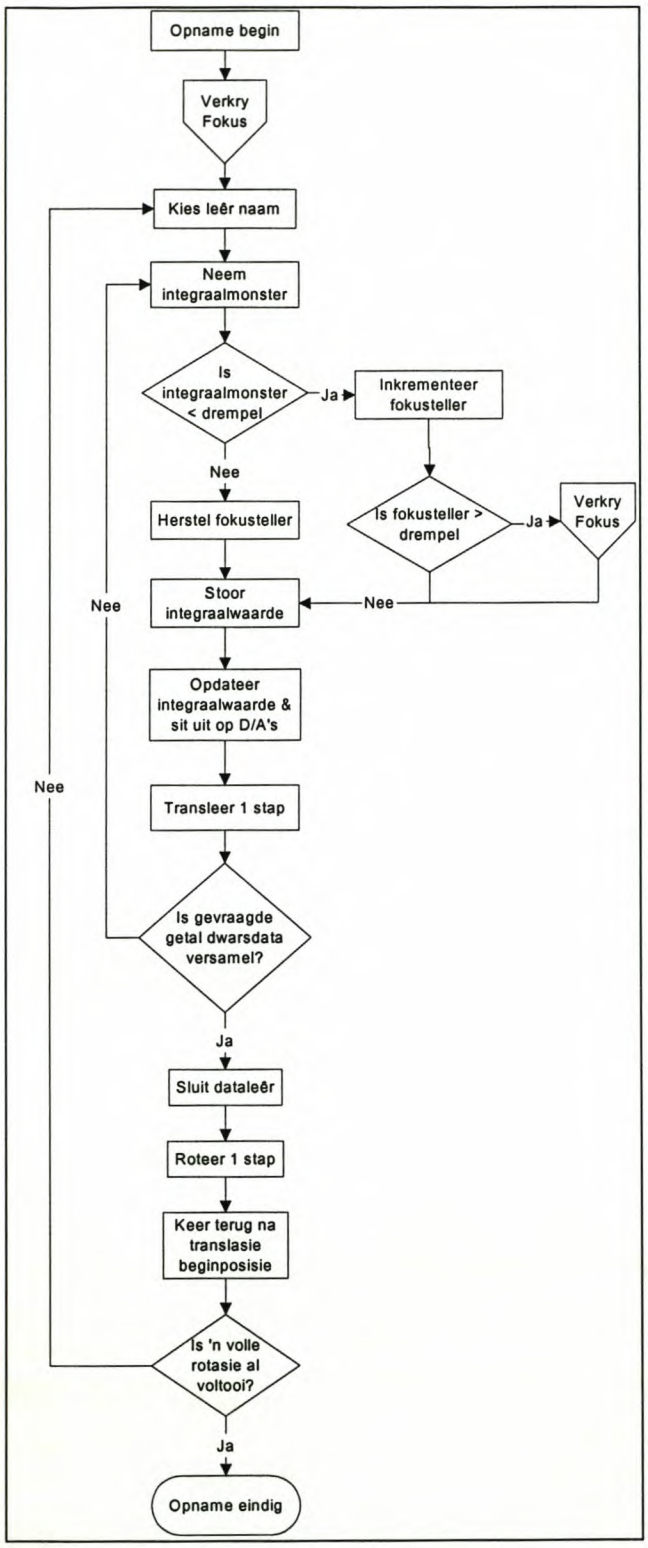


Figuur 31 - Geslotelus frekwensieweergawe van dwarsskandering se terugvoerstelsel.

monsterwaarde $e(k)$ in die algebraïese vergelyking wat die integraal (wat op bladsy 39 getoon word) benader, vermenigvuldig moet word, is dus $K_I \times T = 30 \times \frac{1}{240} \approx \frac{1}{8}$. Met implementering van die algebraïese vergelyking vir die integraal, word $e(k)$ dus deur agt gedeel.

Daar bestaan 'n resolusie-wanaanpassing tussen die D/A's en A/D in die terugvoerstelsel. Die wanaanpassing word veroorsaak deur die reeks aanwinste tussen die integreerderuittree en -intree. Met 'n kwantiseringsbereik van ±10V, is die resolusie van die twee 12-bis D/A's elk $\frac{20}{2^{12}} = 4.88\,\text{mV/bis}$. Na die aanwinste K_{Som} en K_{NN} is die resolusie $4.88 \times 0.11 \times 28 = 15\,\text{mV/bis}$. Die resolusie wat die D/A's by die intree van die A/D lewer, is dus $\frac{15}{4.88} = 3.07$ keer growwer as die resolusie van die A/D. Hierdie wanaanpassing veroorsaak 'n ossillasie teen helfte van die monsterfrekwensie van die integreerder.

Die skematiese diagram van die analoog stroombaan wat gebruik is om die dwarsskanderingmetode te implementeer, word in bylae C getoon. Figuur 32 toon die vloeidiagram wat die werking van die dwarsskanderingprogram illustreer. Die subprosedure "Verkry Fokus" is identies aan die subprosedure vertoon in Figuur 29(b).



Figuur 32 - Vloeidiagram wat werking van die dwarsskanderingprogram illustreer.

Resultate en data van die dwarsskandering word in hoofstuk 6 behandel.

5.7 Gevolgtrekkings

Die voordele wat die dwarsskanderingmetode bo die ander opneemmetodes inhou, is dat daar veel meer inligting is om mee te werk. Die hele groefprofiel kan benut word om tot klank te verwerk.

Indien impulsiewe eksterne steurnisse tydens 'n opname op die sensor inwerk, sal die steurimpuls nie ernstige verwringing aan die data veroorsaak nie, aangesien daar nie saam met die kousaliteit van die klank op die silinder opgeneem word nie. 'n Steurimpuls kan dus hoogstens 'n enkel monster uitskieter in die klanksein veroorsaak wat baie maklik geïdentifiseer en geëlimineer kan word.

Nadele van die dwarsskandering-metode is dat die opneemproses baie lank duur (in die orde van 'n

paar dae). Die datastel kan ook baie groot raak, maar is nogtans veel kleiner as in die geval van die mikroskopiese beeldmetode.

Dit kan ook moontlik wees dat die opnames tussen verskillende dwarslyne nie 'n definitiewe absolute verwysing het nie. Dit beteken dat opeenvolgende dwarslyne met baie klein afsette kan verskil, wat beteken dat daar baie hoë-frekwensie ruis in die klanksein teenwoordig is, maar dit kan egter baie maklik uitgefilter word deur 'n laagdeurlaatfilter.

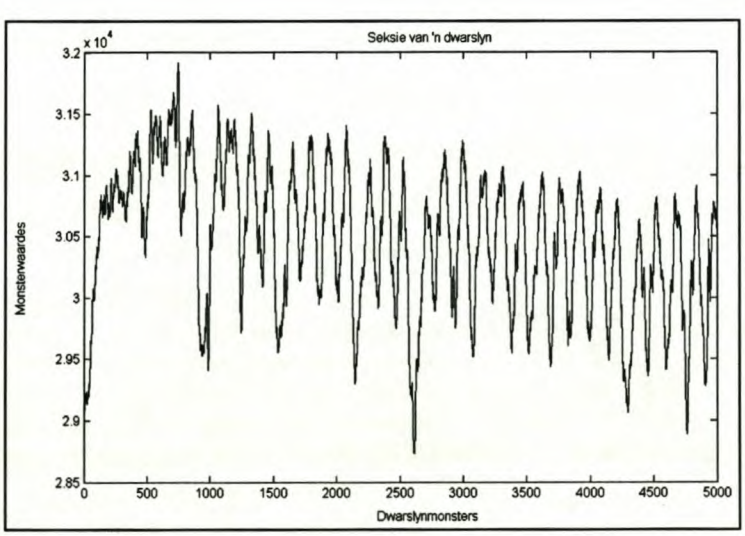
Hoofstuk 6 - Seinverwerking op dwarsskanderingdata

In hierdie hoofstuk word die verwerking van die dwarsskanderingdata tot klank bespreek. Eerstens word die resultate van die dwarsskandering ondersoek, gevolg deur 'n bespreking van die onreëlmatighede teenwoordig in die data met moontlike oplossings. In die afdeling daarna word die metodes wat uitgevoer is om die klank te onttrek, in meer detail behandel.

6.1 Data en resultate van die dwarsskanderingmetode

Figuur 33 toon 'n enkele dwarslyn met monsterwaardes vir 5000 translasiestappe. Hoewel die meeste groefdalle duidelik sigbaar is, is sommige groewe moeilik om te identifiseer. Dit kan aan drie redes toegeskryf word. Eerstens kan baie vlak modulasie moeilik sigbaar wees. Tweedens, weens die V-vormige vorm van die opneemnaald, kerf groewe met baie diep modulasie in aangrensende groewe wat die vlakker groef kan verbloem. Hierdie intermodulasie tussen groewe kan ook steurende eggo's veroorsaak. Derdens beskadig kepe en krapmerke die silinderoppervlak, wat baie duidelik as diep dalle in die dwarslyndata vertoon.

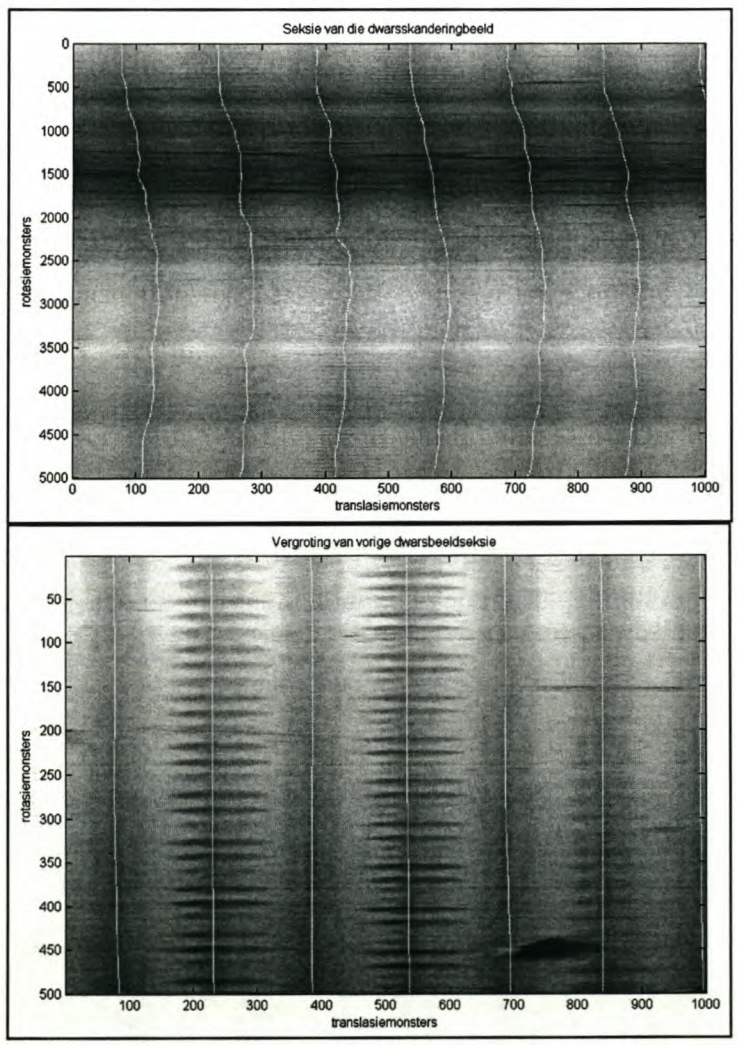
Figuur 34(a) toon die beeld wat gevorm word wanneer 'n hele omwenteling van dwarslyne langs mekaar gerangskik word. Donker streke verteenwoordig lae



Figuur 33 - Seksie van dwarslyndata.

monsterwaardes terwyl ligter streke monsters met hoë waardes verteenwoordig. In hierdie beeld word ses rotasies vertoon waar die ses vertikale wit lyne die pad van die groefmiddelpunt verteenwoordig. Die opneemrigting is van bo na onder op die beeld. Die groef wat dus onder op die beeld by translasiemonster 110 eindig, word bo voortgesit by translasiemonster 225. So vou die groewe op die beeld om, om een aaneenlopende groef te vorm.

Figuur 34(b) is 'n vergroting van 'n seksie van Figuur 34(a). Die dieptemodulasie is baie duidelik sigbaar, asook wydtemodulasie gevorm deur die V-vorm van die opneemnaald. Die beelde stem grootliks ooreen met die mikroskopiese beelde wat in Hoofstuk Drie bespreek is. 'n Baie duidelike keep uit die silinderoppervlak is sigbaar



Figuur 34 - (a) toon die beeld gevorm uit 'n stel aangrensende dwarslyne en (b) toon 'n vergroting van 'n seksie van (a).

onder op die beeld tussen translasiemonsters 700 en 800.

6.2 Onreëlmatighede teenwoordig in die dwarsskanderingdata

Daar bestaan 'n paar ernstige onreëlmatighede wat vanaf die dwarsskanderingdata in Figuur 33, Figuur 34 en Figuur 35 opgemerk kan word. Die vier onreëlmatighede is:

1. Onegalige silindriese vorm
2. Swak klankstrook
3. Laskrake
4. Uitskieters

Die twee ernstigste onreëlmatighede is die onegaligheid in die silindriese vorm van die silinder, en 'n strook van baie swak kwaliteit klank wat oor die lengte van die silinder strek. Krake by gelaste silinders vertoon ook baie duidelik op die beelde deur 'n skielike kleurvlakverandering. Uitskieterdatalyne kom ook soms voor. Die vier onreëlmatighede word in die volgende afdelings bespreek, gevolg deur die metodes wat uitgevoer is om die onreelmatighede te korrigeer.

6.2.1 Onegalige silindriese vorm

'n Geleidelike verandering van ligter na donker skakerings kan in Figuur 34(a) opgemerk word. Hieruit sien ons dat die silinder nie perfek silindries is nie wat veroorsaak dat die silinder onegalig roteer teen die 2.5Hz oorspronklike rotasiefrekwensie. Dit veroorsaak 'n steurende 2.5Hz pulserende ossillasie. Daar bestaan ook 'n helling in die dwarslyn wat duidelik in Figuur 33 sigbaar is.

Hierdie steurnisse is op twee maniere geëlimineer. Die eerste metode maak gebruik van eerste-orde polinome wat op elke dwarslyn gepas word en van die dwarslyn afgetrek word om enige helling en gs-afset van die dwarslyndata te verwyder. Die tweede metode maak gebruik van 'n beeldverwerkingsbeginsel genaamd agtergrondgelykmaking. Die aanname agter hierdie metode is dat die beeldkenmerke van belang se grootte beperk is en op 'n kleiner skaal verander as die onegalige silinderoppervlak. 'n Tweedimensionele orde-statistiese maksimumfilter is gebruik om die groefdalle uit te lig en met die silinderoppervlak gelyk te maak. So word 'n voorgrondbeeld van die onegalige silinderoppervlak verkry. Hierdie voorgrondbeeld

word van die oorspronklike beeld afgetrek om die onegaligheid van die silinderoppervlak te elimineer.

Die implementering en resultate van bogenoemde twee metodes word verder in afdeling 6.3.1 bespreek.

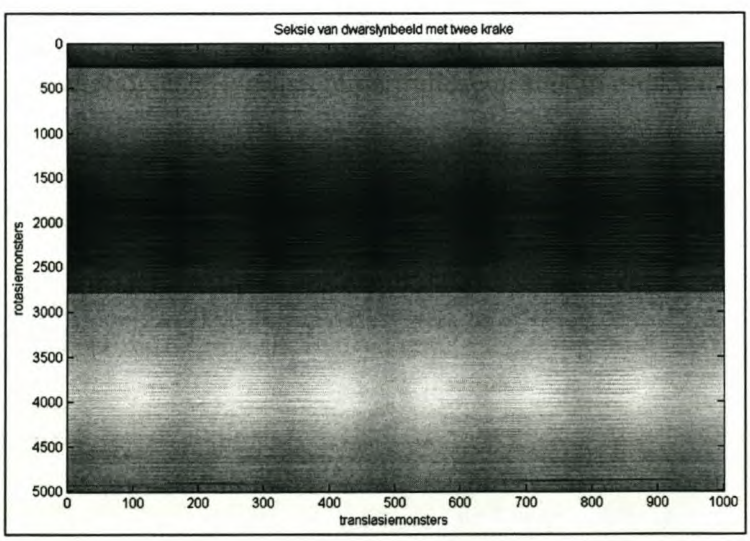
6.2.2 Swak klankstrook

In Figuur 34(a) kan 'n donker strook opgemerk word wat vanaf rotasiemonsters 1000 tot 2000 strek. By nadere ondersoek is gevind dat hierdie gebied 'n strook op die silinder met baie swak gehalte klank met uitermatig baie kepe en krapmerke is. Die beskadigde strook veroorsaak 'n ernstige periodiese gesuis in die klanksein.

'n Moontlike verklaring vir die teenwoordigheid van die swak strook is dat die silinder vir 'n aansienlike tydperk horisontaal op die strook gelê het. As gevolg van die gewig van die silinder op die strook van kontak, het die groewe mettertyd vervlak en beskadig geraak.

6.2.3 Laskrake

In Figuur 35 is twee horisontale krake op die beeld van 'n gelaste silinder sigbaar by rotasiemonsters 250 en 2800. Die onegalige rotasie van die silinder is ook sigbaar. Indien die kraak nagenoeg horisontaal op die beeld is en slegs 'n klein aantal rotasiemonsters insluit (minder as 30 monsters), kan die versteurde dwarslyne bloot



Figuur 35 - Seksie van dwarslynbeeld met twee krake teenwoordig by rotasiemonsters 250 en 2800.

uitgegooi word, sonder enige merkbare verskil aan die klank. Krake wat diagonaal of vertikaal oor die beeld loop was nie teenwoordig by die twee silinders wat tydens die navorsingstydperk ondersoek is nie en is dus nie van toepassing nie.

6.2.4 Uitskieters

In die spesifieke data-opname het gedeeltes van vier dwarslyne buitensporige uitskieterwaardes. Die uitskieterdwarslyne word veroorsaak wanneer die lasersensor langs die rand van die silinder begin monster. Die lasersensor sien die rand van die silinder as 'n baie diep dal met 'n steil helling. Die bandwydte van die sensorterugvoerstelsel laat nie toe dat die integraal hierdie steil helling van die silinder se rand vinnig genoeg volg nie, en neem ongeveer 2500 monsters om bestendigheid te bereik.

Hierdie uitskieterwaardes word as drastiese enkelmonster periodiese klikke in die klanksein waargeneem. Hierdie uitskieters kan gelukkig maklik reeds in die dwarslynbeeld met 'n tweedimensionele mediaanfilter gekorrigeer word.

6.3 Korreksie van onreëlmatighede

Die metodes wat toegepas is om bogenoemde onreëlmatighede te korrigeer, word in hierdie afdeling behandel.

6.3.1 Onegalige silindriese vorm

Die onegalige vorm van die silinder veroorsaak 'n steurende 2.5Hz pulserende ossillasie. Twee maniere is ondersoek om die onegaligheid te elimineer. Eerstens is 'n lynpassingmetode probeer en tweedens 'n beeldverwerkingsmetode vir agtergrond-gelykmaking.

Die lynpassingmetode word in detail in die volgende paragrawe bespreek. Die 'polyfit'-funksie in Matlab is gebruik om die helling en afset van die dwarslyn te karakteriseer. Die helling- en afsnitparameters vir elke dwarslyn is in twee vektore van 5000 elemente elk gestoor. Indien die lynpassingsvektor van die dwarslynvektor afgetrek word, word die dwarslyn se hellingsteurnis en gs-afset dus geëlimineer.

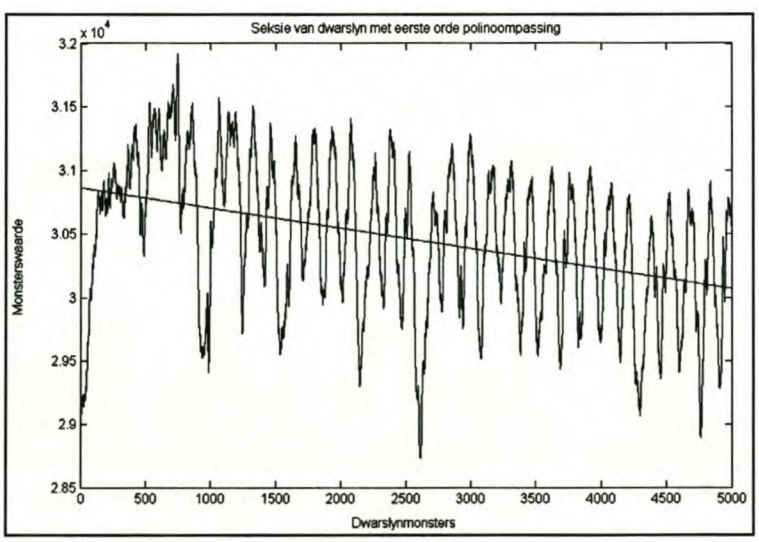
Hierdie lynpassingmetode is nie objektief ten opsige van die dwarslyn waarop dit uitgevoer word nie. 'n Dwarslyn met diep groefdalle gevorm deur diep modulasie of 'n keep, sal die lynpassing tot 'n laer afset dwing. Die lynpassing se afset dwing so ongewenste vervormingseffekte op ander groefdalle van dieselfde dwarslyn af. 'n Diep groefdal kan 'n skerper helling op die lynpassing afdwing, wat weer 'n ongewenste vervlakking of verdieping van nabyliggende groefdalle veroorsaak. Die lynpassings veroorsaak dus dat groewe wat meer aansienlike modulasie besit, hul klank op naasliggende groewe kan afdwing, wat steureggo's teen 2.5Hz veroorsaak.

Die lokale kenmerke wat die dominerende groefdalle op naasliggende groewe afdwing moet uitgefilter word om 'n meer algemene lynpassing te verkry, om sodoende die 2.5Hz steureggo's te minimeer. Die helling- en afsnitparametervektore is met 'n bewegende gemiddeldefilter gefilter om die parameters te verglad. Die bewegende gemiddeldefilter besit 'n oordragsfunksie wat soos volg lyk:

$$H(z) = \frac{1}{N}\sum_{i=1}^{N} z^{-i+1},$$

waar N die orde van die filter is.

Die filter is nie-kousaal toegepas deur van die Matlab se 'filtfilt'-funksie gebruik te maak, wat die filter se faseskuif elimineer. 'n Bewegende gemiddelde filter met 'n bandwydte van 250Hz is voldoende om die lokale spraakkenmerke vervat in die lynpassingparameters uit te filter, aangesien die bandwydte van die klank



Figuur 36 - Dwarslyn met eerste-orde polinoompassing.

$250-4000\text{Hz}$ is [2].

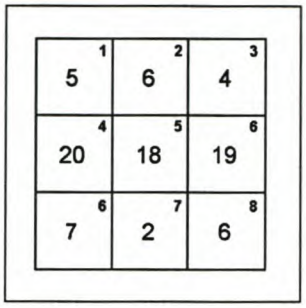
Die nadeel wat hierdie filtering inhou, is dat dit vergladding by die kraaklaste veroorsaak. Juis hier word die karakterisering van die drastiese sprong in die lasvlakke verlang sodat die lynpassing die twee gelaste vlakke tot op 'n gelyke vlak kan bring om sodoende die kraak te elimineer. Die gemiddeldefilter veroorsaak 'n baie gladde oorgang by die kraaklas wat veroorsaak dat die drastiese lasvlaksprong nie behoorlik gelykgemaak word nie. Dit veroorsaak dat daar 'n steurende periodiese spykerpuls in die klank voorkom.

Waardes vir $N>15$ vir die bewegende gemiddelde filter het die steureggo's suksesvol geëlimineer, maar 'n kompromis moet tussen die spykerpulsgrootte en hoeveelheid steureggo gevind word.

In silinders waar geen krake voorkom nie sal hierdie steurende spykerpuls nie voorkom nie. Hierdie metode is dus voldoende om die onegalige silindriese vorm van kraaklose silinders te verwyder.

Die tweede metode wat toegepas is om die silinder se onegalige vervorming te elimineer, is deur middel van agtergronggelykmaking [13]. 'n Orde-statistiese maksimumfilter is gebruik om die groefkenmerke van die dwarsskanderingbeelde te vervaag totdat slegs 'n voorgrondbeeld van die oppervlak van die silinder verkry is. Hierdie voorgrondbeeld kan nou van die oorspronklike beeld afgetrek word om die onegalige vervorming van die silinder te elimineer.

Die werking van die orde-statistiese maksimumfilter kan na aanleiding van Figuur 37 verduidelik word. Die middelste pixelwaarde in 'n blokkie met 'n gepaste grootte, word vervang met die maksimum waarde van die pixels teenwoordig in die blokkie. Pixel vyf in Figuur 37 sal dus met die grootste waarde teenwoordig in die 3×3 blokkie vervang word, naamlik 20. Die 'ordfilt2'-funksie van Matlab is gebruik om die filter te realiseer.



Figuur 37 - Filterblokkie

Die filterblokkie moet ten minste die hele groefbreedte dek om die groefkenmerke te vervaag. Die maksimumwaarde teenwoordig in die blokkie stel vermoedelik die silinderoppervlak voor.

Die lengte van die blokkie is klein gekies om te voorkom dat positiewe uitskieterwaardes nie bevoordeel word en op die beeld "groei" nie. Aangesien die breedte van die groef 152 monsters is, is 'n gepaste grootte vir die filterblokkie 3×152.

Na toepassing van die maksimumfilter, het die voorgrondbeeld skerp oorgange tussen aangrensende groewe. Dit word veroorsaak deur die verskil tussen die maksimum waardes tussen die streke wat deur die filterblokkie gedek word. Om hierdie verskynsel te onderdruk, word 'n tweedimensionele gemiddelde filter op die voorgrondbeeld toegepas om die beeld verder te verglad. Resultate en figure word in bylae D getoon.

6.3.2 Laskrake

Die maklikste en effektiefste metode om horisontale laskrake te elimineer, is deur die groep korrupte dwarslyne bloot te verwyder, mits slegs 'n klein aantal rotasiemonsters (ongeveer 30) versteur is. Daarna kan metodes vir die verwydering van die onegalige silindriese vorm op die data toegepas word.

6.3.3 Uitskieters

Uitskieterdwarslyne word maklik met 'n tweedimensionele mediaanfilter gekorrigeer. Die tweedimensionele mediaanfilter is ook 'n orde-statistiese filter, soos die maksimumfilter. Die middelste pixelwaarde in 'n blokkie van gepaste grootte, word vervang met die mediaanwaarde van die pixels teenwoordig in die blokkie [13]. Pixel

vyf in Figuur 37 sal dus met die mediaanwaarde teenwoordig in die 3×3 blokkie vervang word, naamlik ses. Die 'medfilt2'-funksie van Matlab is gebruik om die filter te realiseer.

Die mediaanfilter is slegs op 'n klein streek rondom die posisie waar 'n uitskieterdwarslyn voorkom toegepas, om te verhoed dat die hele beeld se data nie deur die filter beïnvloed word nie. Die grootte van die mediaanfilterblokkie moet ook so gekies word dat dit nie die korrekte datawaardes korrupteer nie. Aangesien die uitskieters slegs enkele dwarslyne, omring deur korrekte data, is, kan die kleinste moontlike filterblokkie, naamlik 3×3, gebruik word om die uitskieters te korrigeer.

6.3.4 Swak klankstrook

Die eenvoudigste en doeltreffendste oplossing wat toegepas is om die periodiese suis van die swak klankstrook te onderdruk, is om die strook te vermenigvuldig met 'n funksie wat bloot verswakking van die suis veroorsaak. 'n Bevredigende resultaat is verkry nadat die stook met 'n verhewe kosinus met 30% verswakking vermenigvuldig is. Die verswakkingsfunksie is toegepas op die beeld vanaf rotasiemonsters 700 tot 2300 oor die lengte van die beeld.

'n Ander oplossing is ook in die tydgebied uitgevoer, waar die verwronge aantal monsters met 'n interpolasie vervang word deur 'n afskatting vanuit die naasliggende korrekte monsters te maak. Hierdie metode word in detail in afdeling 7.1 bespreek.

6.4 Klankonttrekking

Eerstens is die groefsteek verwyder om die rotasiegroefie nagenoeg vertikaal op die dwarsskanderingbeeld te kry. Tweedens is die groefpad vir elke rotasiegroef op die dwarsskanderingbeeld bepaal. Derdens word 'n profiel van die groef op 'n vaste breedte om die groefpad geneem. Hierdie profiel moet dan tot klankmonsters verwerk word.

6.4.1 Verwydering van die groefsteek

Wat tot nou toe nog nie genoem of vertoon is nie, is dat die steek van die groef ook duidelik op die dwarsskanderingbeelde sigbaar is. Dit is sigbaar deurdat die groewe

op die dwarsskanderingbeeld skuins van links na regs loop en dat die een groefrotasie eindig by die translasiemonster waar die volgende groefrotasie op die beeld begin.

Die steek van die groewe word verwyder om die onderskeie rotasiegroewe nagenoeg vertikaal te kry. Die onderskeie groewe kan nou maklik in smal snitte afgesonder word. Die Viterbi-algoritme word op elke individuele rotasiegroef uitgevoer om die mees waarskynlike groefpad te vind. Die breedte van die snit, in translasiemonsters, verteenwoordig die aantal Markov-toestande M waaroor die algoritme uitgevoer moet word. Aangesien die verwerkingstyd van die algoritme kwadraties van die aantal toestande afhang, is dit van belang dat die snit waarop die algoritme uitgevoer word so smal as moontlik moet wees om verwerkingstyd te minimeer.

Deur die steek van die groewe te verwyder, is slegs een groef in 'n snit teenwoordig. Indien die steek van die groef nog teenwoordig sou wees, en 'n groef word in 'n snit afgesonder, word die ente van aangrensende groewe ook in die snit ingesluit. Die datawaardes van hierdie aangrensende groewe kan veroorsaak dat die Viterbi-algoritme die groefpad foutief bepaal, aangesien die aangrensende groewe se datawaardes ook as toestandswaarskynlikhede in berekening gebring word. 'n Ander nadeel is dat 'n breër snit afgesonder moet word om die groef in te sluit, wat meer Markov-toestande beteken, wat meer verwerkingstyd opneem.

Verwydering van die groefsteek is gedoen deur 'n aantal monsters aan die linkerkant van die dwarsskanderingbeeld te verwyder na aanleiding van die rotasiemonsterposisie. Die steek van die groef, in aantal monsters, soos bereken in afdeling 3.2.2 , is 32.8084 rotasiesmonsters vir elke translasiemonster. Dit kan afgerond word na 33. Die groefsteek word verwyder deur met elke faktor van 33 van die rotasiemonsters nog 'n translasiemonster aan die begin (linkerkant) van die dwarslyn te verwyder.

In Figuur 34 en Figuur 35 is die steek van die groef reeds verwyder. Noudat die groewe nagenoeg vertikaal op die beeld is, kan individuele rotasiegroewe maklik in aparte snitte afgesonder word. Die Viterbi-algoritme word op hierdie snitte toegepas.

6.4.2 Viterbi-algoritme om die groefpad te bepaal

Die Viterbi-algoritme is gebruik om die mees waarskynlike groefpad te bepaal. Die algoritme is individueel op elke rotasiegroef van die dwarsskanderingbeeld uitgevoer tussen vasgestelde begin- en eindtoestande. Die translasiemonsters dien as die Markov-toestande en die rotasiemonsters as die tydstappe vir die algoritme. Die klokvormige cosinus-funksie word weereens as die oorgangswaarskynlikhede a_{ij} gebruik soos in Afdeling 4.8 bespreek. Die negatief van die dwarslyndatawaardes dien as die toestandskoste b_i , aangesien die omgekeerde groefdal 'n piek word wat die mees waarskynlike groefmiddelpunt voorstel.

Die groefdalposisies aan die begin en einde van elke groefrotasie is as die begin- en eindtoestande vir die Viterbi-algoritme vir elke groefsnit gebruik. 'n Eenvoudige dalposisie-algoritme is geïmplementeer om die groefdalposisies te vind. Die dalposisie-algoritme is op 'n gemiddelde dwarslyn toegepas, aangesien 'n enkele dwarslyn se groefprofiel growwe data besit en van die groefdalle baie vlak kan wees. Die gemiddelde dwarslyn is verkry deur die gemiddeld van 20 aangrensende dwarslyne aan die begin en einde van die beeld te neem. Hierdie gemiddelde dwarslyn toon 'n baie gladde groefprofiel, waarvan selfs baie vlak groefdalle duidelik sigbaar is.

Die gemiddelde dwarslyn word gefilter met die volgende oordragsfunksie:

$$H(z)=\frac{1}{N}\sum_{i=0}^{N-1}(-z^{-i}+z^{-i-N}) \quad \text{met } N=30.$$

Hierdie filter is identies aan die vergladde afgeleide filter wat in Afdeling 4.8 gebruik is om die toestandskostefunksie vanaf die beelddata te genereer.

Die gefilterde data se nulkruisings met negatiewe hellings, toon die monsterposisies waar dalle in die gemiddelde dwarslyn voorkom, met 'n fasevertraging van $2N$ monsters.

Elke rotasiegroef word onderskeidelik in snitte afgesonder waarop die Viterbi-algoritme toegpas word. Die breedte van die groefsnit, in translasiemonsters, wat die aantal Markov-toestande verteenwoordig, is gekies as $M=100$. Vir hierdie waarde is die snit breed genoeg om al die groefdalle van die spesifieke rotasiegroef in te sluit,

aangesien die steek van die groef verwyder is. Die waarde is ook klein genoeg om verwerkingstyd te minimeer.

Die aantal tydstappe vir die Viterbi-algoritme is $K = 5000$ aangesien daar 5000 monsterpunte per rotasie is. Die beperking vir die klokvormige oorgangswaarskynlikheidsverspreiding a_{ij} is gekies as nege, wat toelaat dat 'n maksimum toestandsprong van sewe toestande tussen twee tydstappe kan voorkom.

Die gevolglike groefpadkontoere besit redelik hoë-frekwensie spronge soos die toestandsoorgange tussen tydstappe met tot 7 toestande kan wissel. Die groefpadkontoer word deur 'n bewegende gemiddelde filter gestuur, om 'n gladder kontoer te verkry. Die oordragfunksie van die filter lyk as volg:

$$H(z) = \frac{1}{N}\sum_{i=1}^{N} z^{-i+1} \qquad \text{waar } N = 200 .$$

Die nie-kousale 'filtfilt'-funksie van Matlab, wat die filter se faseskuif elimineer, is vir die doel aangewend. Die gladde kontoere word in Figuur 34(a) en (b) getoon.

Die groefpadkontoer dien as indeks waar die klankmonsters op die beeld teenwoordig is. Klank word verkry deur die monsterwaardes by hierdie posisies in 'n datary te plaas. Elke groefrotasie se datawaardes word in die korrekte volgorde agtermekaar gelas om een lang datary van monsterpunte te vorm. Voordat die klank uit die groefpadkontoere onttrek word, moet die onderskeie onreëlmatighede wat vroeër in die hoofstuk bespreek is, eers uit die weggeruim word.

6.4.3 Verwerking van die groefprofiel

Die klank van 'n enkele groefpad is hoorbaar maar baie ruiserig en nie duidelik verstaanbaar nie. Dit sou 'n vermorsing van inligting wees om klank slegs uit een kontoer in die groef te herwin aangesien daar nog baie bruikbare inligting in die wande van die groefprofiel teenwoordig is. Dit is dus meer sinvol om 'n hele profiel van die groef in ag te neem en klank daaruit te onttrek.

'n Profiel van die groef kan verkry word deur 'n aantal monsterpunte rondom die mees waarskynlike groefpadkontoer te neem. 'n Metode moet dus gevind word om die groefprofiel, wat uit 'n aantal monsterpunte bestaan, tot 'n klankmonster te verenig.

Daar is gevind, deur na die verskillende enkelpaaie van klank op die groefprofiel te luister, dat die klank nie identies vir verskillende dele in die groefwand is nie. 'n Klankpad wat baie hoog teen die wand van die groef geleë is, kan dus reeds modulasie van die aangrensende groef insluit wat as eggo's gehoor word. Die beste aantal groefpaaie rondom die mees waarskynlike groefpad of –middelpunt moet dus gebruik word om tot klank te verwerk word. 'n Strook van tagtig monsters breed, is rondom die mees waarskynlike groefmiddelpunt geïdentifiseer as die mees bruikbare deel van die groefprofiel.

Twee metodes is ondersoek om die groefprofiel na klank te verwerk. Aangesien die groefprofiel nagenoeg parabolies is, is 'n paraboolpassingsmetode beproef. 'n Eenvoudiger metode wat ook toegepas is, is om slegs die gemiddelde waarde van die groefprofiel as die klankmonster te neem.

'n Tweede-orde polinoomfunksie is op die groefprofiel gepas deur van Matlab se 'polyfit'-funksie gebruik te maak. Die funksie gebruik die kleinste-kwadraat-metode om die polinoompassing te kry. Die draaipunt van die parabool word dan as klankmonster gebruik.

Die draaipunt kan nie altyd as die korrekte monsterwaarde aanvaar word nie. 'n Baie vlak groef kan dele van aangrensende groewe se wande in sy profielsnit bevat wat die paraboolpassing onbetroubaar maak. Krake en kepe verwring ook die groefprofiel en veroorsaak dat die paraboolpassing onbetroubare en buitensporige resultate lewer deurdat die draaipunt van die parabool 'n uiterste waarde wyd buite die grense van die minimum of maksimum van die datastel, besit.

'n Poging is aangewend om 'n parabool met 'n vaste vorm op die profiel te pas. Die stel matriks vergelykings vir die kleinste-kwadraat-oplossing in die 'polyfit'-funksie, is aangepas om 'n parabool met 'n vaste vorm (konstante faktor vir die kwadratiese term) op die groefprofiel te pas. Veel beter resultate is behaal as die oorspronklike paraboolpassingsmetode.

Die eenvoudigste metode om die groefprofiel tot 'n enkele monsterpunt te verenig, is om die klankmonsters uit die gemiddelde waarde van die groefprofiel te skep. Die voordeel wat die gemiddelde groefprofielwaarde inhou, is dat 'n verwronge groefprofiel nie uiterste klankmonsterwaardes kan voortbring wat buite die grense van

die minimum- en maksimumwaarde van die groefprofiel lê nie, soos in die geval van die paraboolpassingsmetode.

Tussen die gemiddelde waarde van die groefprofiel en die paraboolpassingmetode kon daar geen hoorbare verskil tussen die twee klanksnitte waargeneem word nie. Daar is om hierdie rede besluit om die gemiddelde-waarde-metode te gebruik, aangesien dit eenvoudig is en vinnig verwerk.

6.5 Gevolgtrekkings

Die klanksnit wat verkry is deur die gemiddeld van die groefprofiel te bereken, word in Afdeling 7.4 , Figuur 46(a), getoon. Die kepe en krapmerke, teenwoordig veral as gevolg van die swak klankstrook, vertoon veel groter as in die geval met die voorwaartse opneemmetode. Die rede hiervoor is dat die klanksein nog geen vorm van filtering of verwerking ondergaan het nie en die rouste moontlike klank beskikbaar is vanaf die silinder. Die verwringing in die dwarsskanderingklanksnit vertoon as diep uitbarstings wat vermiste dele in die klanksnit is, aangesien daar geen spraakinligting onderliggend aan hierdie tipe uitbarstingsruis is nie. Die uitbarstings is duidelik sigbaar en in hoofstuk 7 word 'n restourasie-algoritme bespreek om hierdie vermiste klanksegmente met beter interpolasies te vervang.

Die bandbeperking van die voortwaartse opneemmetode maak die ruis veel meer gekleurd en gekorreleerd met die spraak. Die SNR van die voorwaartse opneemmetode is aanvanklik ietwat beter, maar dit is veel moeiliker om in verdere verwerking die ruis van die klank te skei of onderskei.

Die SNR vir 'n enkele kontoerklanksnit word vergelyk met dié van die gemiddelde groefprofielklanksnit. Die SNR is bereken deur dieselfde prosedure van Afdeling 5.6.4 te volg. Die gemiddelde SNR vir die 80 enkelkontoerklanksnitte is bereken as 0.52dB. Die SNR vir die gemiddelde groefprofielklanksnit is bereken as 2.77dB. Oudiorestorasie en twee spraakverheffingsmetodes word in die volgende hoofstuk ondersoek.

Hoofstuk 7 - Digitale restourasie en spraakverheffing

'n Handoudiorestourasieprogram is in Matlab geskryf om onder meer vermiste klanksegmente en sigbare verwringing in die klanksein met beter interpolasies te vervang. Die verwringing en vermiste klanksegmente is grotendeels afkomstig van diep kepe, krapmerke, laskrake en die swak klankstrook (afdeling 6.2.2) op silinders.

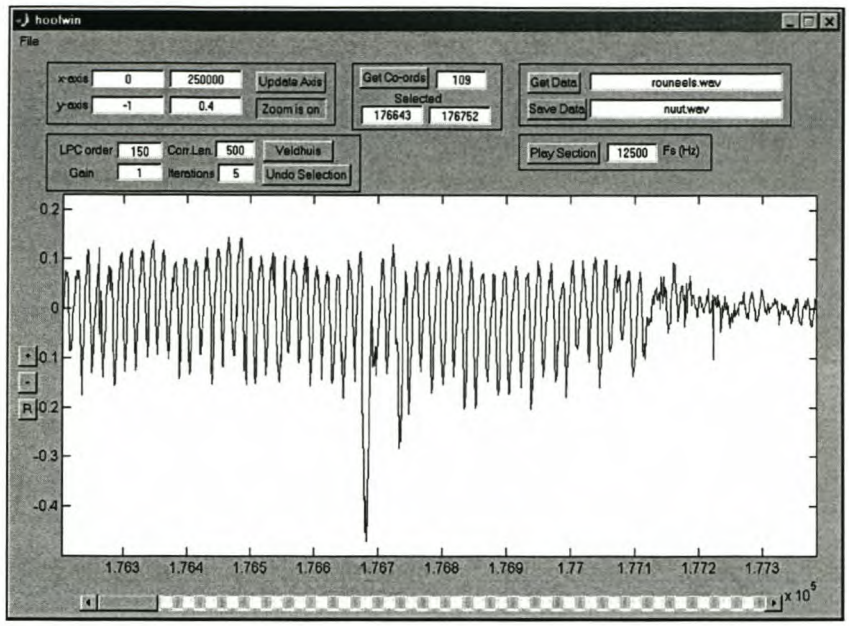
Twee metodes van spraakverheffing is op die rou klanksnit toegepas. Die eerste metode van spraakverheffing wat ondersoek is, is STSA ('short-time spectral attenuation'). Dit is die gewildste metode wat deur kundiges gebruik word om klankopnames van verouderde oudiomedia te verbeter. Die nuutste verwikkelinge in STSA is om ook die psigoakoestiese waarnemingseienskappe van die menslike gehoor met die verheffing van die verwronge klanksein in ag te neem. Die voordeel hiervan is 'n vermindering van die sogenaamde musikale ruistone wat deur STSA veroorsaak kan word, asook verheffing van baie sagte klanksegmente.

'n Tweede, veel eenvoudiger metode wat vir spraakverheffing gebruik is, is om bloot vir elke korttydraampie lineêre voorspellingskoëffisiënte af te skat en hierdie LPC-filter (lineêre voorspellingskoëffisiënte) op die raampie toe te pas om sodoende die spraakmodel te verhef.

In hierdie hoofstuk word die oudiorestourasieprogram eerstens bespreek, gevolg deur die twee spraakverheffingsmetodes. Die twee spraakverheffingsmetodes is deur luistertoetse wat deur 16 luisteraars geëvalueer is, teen mekaar opgeweeg.

7.1 Handoudiorestourasieprogram

'n Oudiorestourasieprogram is in Matlab geskryf. Duidelik sigbare verwronge monsters, wat met sig en gehoor geïdentifiseer kan word, kan met 'n beter interpolasie vervang word. Die interpolasie is 'n lineêre minimum-variansie afskatting, wat uit die AR-parameters (outoregresiewe parameters) van die omliggende onverwronge monsters bereken word. Die hoëvlakwerking van die restourasieprogram word beskryf, gevolg deur 'n bespreking van die laevlakwerking (restourasie-algoritme).



Figuur 38 - Handoudiorestorasieprogram

7.1.1 Hoëvlakwerking van die oudiorestourasieprogram

Die oudiorestourasieprogram se koppelvlak is ten volle grafies en interaktief en word in Figuur 38 getoon. 'n Windows ".wav"-lêer kan gelaai word en die tyddata van die klanksnit word op 'n assestelsel vertoon. Korrupte pulse kan maklik met die oog geïdentifiseer word, deur op seksies van die klanksnit in te fokus. Dit is ook moontlik om na die klankseksie wat in die assestelsel vertoon is, te luister.

'n Korrupte dataseksie word met die muis geselekteer en met 'n beter interpolasie vervang. Verstelbare parameters wat die interpolasie beïnvloed, is die orde van die outoregresiewe model, die aantal korrekte monsters wat in ag geneem moet word rondom die korrupte seksie, asook die aantal iterasies wat uitgevoer moet word. Indien 'n interpolasie onbevredigend voorkom, kan die parameters verstel word tot 'n beter interpolasie verkry is. Die gekorrupteerde dataseksie kan ook weer terug vervang word. Die verwerkte klanksnit word as 'n nuwe ".wav"-lêer gestoor.

7.1.2 Laevlakwerking van die oudiorestourasieprogram

Die restourasie algoritme is 'n toepassing van Hoofstuk Drie in [14]. Die algoritme is 'n veralgemeende AR-voorstelling wat nie-korrupte monsterpunte rondom 'n bekende korrupte monsterposisie in ag neem om 'n lineêre minimum-variansie afskatting vir die korrupte monster te vind. Die afskatting is lineêr omdat die waardes van die onbekende monsters afgeskat word as 'n lineêre kombinasie van die omliggende

bekende monsters. Dit is ook 'n minimum-variansie afskatting omdat die variansie van die afskattingsfout geminimeer word.

Die aannames wat in verband met die sein onder restourasie gemaak word, is dat die sein 'n realisering van 'n tweede-orde stasionêre stochastiese proses $\underline{s}_i$ is, wat as 'n outoregressiewe proses gemodelleer kan word. Dit is voldoende om te aanvaar dat 'n spraaksein tweede-orde stasionêr oor 'n kort tydraampie van 20-40ms is [14].

Laat die monsters van die kort tydraampie van lengte N voorgestel word deur

$$s_k\,,\ k=1,\ldots,N\,.$$

Hierdie kort tydraampie bevat m onbekende of korrupte monsters by bekende posisies

$$t(i)\,,\ i=1,\ldots,m\,.$$

Laat V, W en $W\setminus V$ onderskeidelik die versamelings van onbekende, alle (bekende en onbekende) en bekende monsterposisies voorstel. Wegingskoëffisiënte

$$h_{ij}\,,\ i=1,\ldots,m\,,\ j\in W\setminus V$$

moet nou gevind word sodat $s_{t(1)},\ldots,s_{t(m)}$ beraam kan word as

$$\hat{s}_{t(i)}=\sum_{j\in W\setminus V}h_{ij}s_j\ ,\ i=1,\ldots,m\,.$$

Figuur 39 toon 'n diagrammatiese voorstelling van hoe die afskatting van onbekende monsters uit die weging vanuit bekende monsters bepaal word.

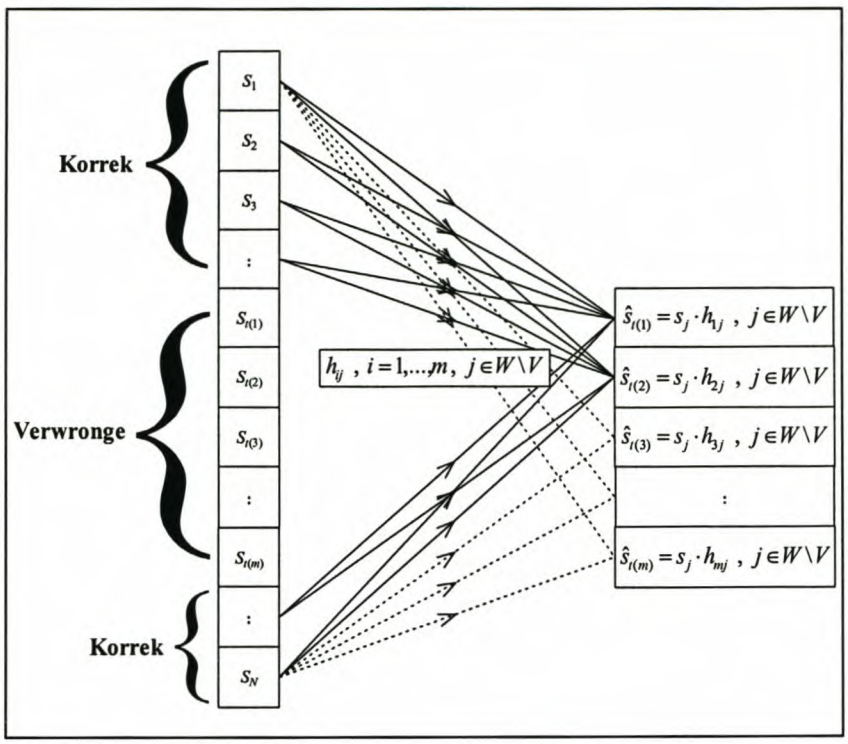
'n Oplossing kan gevind word deur die algehele variansie van die statistiese restourasie fout te minimeer:

$$\mathrm{E}\left\{\sum_{i=1}^{m}\left(\hat{\underline{s}}_{t(i)}-\underline{s}_{t(i)}\right)^2\right\}.$$

Dit word gedoen deur die afgeleide gelyk aan nul te stel om die optimale oplossing te vind. Die oplossing lewer 'n versameling van m stelle vergelykings met $N-m$ veranderlikes en sien soos volg daar uit:

$$\sum_{j\in W\setminus V}h_{ij}R(k-j)-R\big(t(i)-k\big)=0\,,\ k\in W\setminus V\,,\ i=1,\ldots,m\,.$$

Die wegingskoëffisiënte h_{ij} vir m korrupte monsterposisies kan nou opgelos word vanuit m stelle Wiener-Hopf-tipe vergelykings (genommer deur i), indien die outokorrelasiematriks $R(\cdot)$ van die kort tydraampie bekend is. Sien Figuur 40 vir 'n matriksvoortstelling van die m stelle vergelykings.



Figuur 39 - Diagram wat die weging van die korrekte monsters voorstel, om afskattings vir die onbekende monsters te verkry.

Hierdie vergelykings word egter uitgebrei om meer bruikbare en insiggewende verbande met die outokorrelasie te lewer. Vanuit hierdie uitgebreide vergelykings word getoon dat afskattings vir m onbekende monsters gevind kan word deur 'n stel van m vergelykings met m onbekendes op te los. Die metode word verder uitgebrei sodat hierdie stel vergelykings direk van 'n stel AR-parameters, wat die spektrum van die kort tydraampie voorstel, opgelos kan word. Dit is veel meer doeltreffend as om die wegingskoëffisiënte vir elk van die m onbekende monsterpunte af te skat.

By die afleiding van die algoritme word deurgaans aanvaar dat die parameters, voorspellingkoëffisiënte en orde van die AR-proses bekend is. Dit is natuurlik nie die geval in die praktyk nie, en beide die AR-parameters en onbekende monsterwaardes moet van beskikbare, onvolledige data beraam word. Hiervoor is 'n iteratiewe metode, sterk verwant aan die EM-algoritme, afgelei [14].

Die werking van die algoritme word in Figuur 41 vertoon. 'n Korrupte klankseksie word met die muis op die assestelsel geselekteer. 'n Vektor $\mathbf{s}$ met lengte N word geskep. Die vektor lyk soos volg:

$$\mathbf{s} = [(p+1 \text{ korrekte monsters}) \ (m \text{ geselekteerde korrupte monsters}) \ (p+1 \text{ korrekte monsters})]^T$$

waar p die orde van die AR-afskatting is. Die lengte van $\mathbf{s}$ is dus $N = 2(p+1)+m$.

Die keuse van p is ongelukkig heuristies en kan moontlik gemotiveer word deur intuïtief te redeneer. Dit maak sin om die aantal korrekte datapunte ten minste twee keer meer as die aantal onbekende datapunte te maak, aangesien p monsters aan beide kante van 'n aantal onbekende monsters lineêr gekombineer word om 'n oplossing te verkry.

Vektor $\hat{\mathbf{x}} = \left[\hat{s}_{t(1)}\ \hat{s}_{t(2)} \ldots\ \hat{s}_{t(m)}\right]^{\mathrm{T}}$, die oplossing van die onbekende monsters, word opgelos deur:

Laat $N = 8$ en $t(i) = 4,5$ met $i = 1,2$, dus is $m = 2$.

$$\mathbf{R}_{(\text{bekende monsters})}\mathbf{H}^{\mathbf{T}}_{(\text{bekende monsters})} - \mathbf{r}_{t(i),5} = \mathbf{0}$$

$$\begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & r_{1,6} & r_{1,7} & r_{1,8} \\ r_{2,1} & r_{2,2} & r_{2,3} & r_{2,6} & r_{2,7} & r_{2,8} \\ r_{3,1} & r_{3,2} & r_{3,3} & r_{3,6} & r_{3,7} & r_{3,8} \\ r_{6,1} & r_{6,2} & r_{6,3} & r_{6,6} & r_{6,7} & r_{6,8} \\ r_{7,1} & r_{7,2} & r_{7,3} & r_{7,6} & r_{7,7} & r_{7,8} \\ r_{8,1} & r_{8,2} & r_{8,3} & r_{8,6} & r_{8,7} & r_{8,8} \end{bmatrix} \begin{bmatrix} h_{1,1} & h_{2,1} \\ h_{1,2} & h_{2,2} \\ h_{1,3} & h_{2,3} \\ h_{1,6} & h_{2,6} \\ h_{1,7} & h_{2,7} \\ h_{1,8} & h_{2,8} \end{bmatrix} - \begin{bmatrix} r_{1,4} & r_{1,5} \\ r_{2,4} & r_{2,5} \\ r_{3,4} & r_{3,5} \\ r_{6,4} & r_{6,5} \\ r_{7,4} & r_{7,5} \\ r_{8,4} & r_{8,5} \end{bmatrix} = \mathbf{0}$$

$$\begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} & r_{1,5} & r_{1,6} & r_{1,7} & r_{1,8} \\ r_{2,1} & r_{2,2} & r_{2,3} & r_{2,4} & r_{2,5} & r_{2,6} & r_{2,7} & r_{2,8} \\ r_{3,1} & r_{3,2} & r_{3,3} & r_{3,4} & r_{3,5} & r_{3,6} & r_{3,7} & r_{3,8} \\ r_{6,1} & r_{6,2} & r_{6,3} & r_{6,4} & r_{6,5} & r_{6,6} & r_{6,7} & r_{6,8} \\ r_{7,1} & r_{7,2} & r_{7,3} & r_{7,4} & r_{7,5} & r_{7,6} & r_{7,7} & r_{7,8} \\ r_{8,1} & r_{8,2} & r_{8,3} & r_{8,4} & r_{8,5} & r_{8,6} & r_{8,7} & r_{8,8} \end{bmatrix} \begin{bmatrix} h_{1,1} & h_{2,1} \\ h_{1,2} & h_{2,2} \\ h_{1,3} & h_{2,3} \\ -1 & 0 \\ 0 & -1 \\ h_{1,6} & h_{2,6} \\ h_{1,7} & h_{2,7} \\ h_{1,8} & h_{2,8} \end{bmatrix} = \mathbf{0}$$

$$\mathbf{R}'\mathbf{H}^{\mathbf{T}} = \mathbf{0}$$

Waar $\mathbf{R}'$ 'n gereduseerde outokorrelasie matriks van die sein is, en $\mathbf{H}^{\mathbf{T}}$ 'n uitgebreide koëffisiëntmatriks is.

Figuur 40 – Matriksvoorstelling van die Wiener-Hopf-tipe vergelykings.

$$\tilde{\mathbf{G}}\hat{\mathbf{x}} = -\mathbf{z}$$

waar $\tilde{g}_{ij} = b_{t(j)-t(i)},\ i,j = 1,\ldots,m$,

en $z_i = \sum_{k=-p}^{p} b_k v_{k+t(i)},\ i = 1,\ldots,m$.

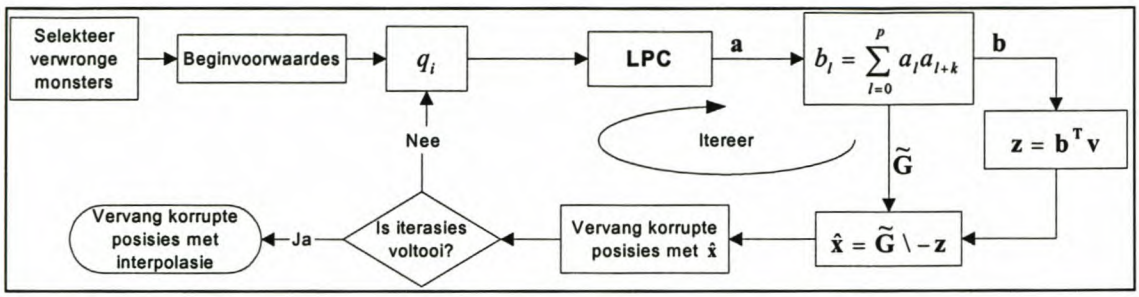
Vektor $\mathbf{z}$ word die sindroom genoem. Die parameters b_k word bereken deur die outokorrelasie van die AR-parameters (a_k) te neem:

$$b_l = \sum_{l=0}^{p} a_l a_{l+k}\ .$$

Vektor $\mathbf{v}$ word gedefinieer as vektor $\mathbf{s}$ met $s_{t(i)} = 0,\ i = 1,\ldots,m$.

Vir die eerste iterasie word $\hat{\mathbf{x}}^{(0)} = \mathbf{0}$ gestel. Die AR-parameters $\hat{\mathbf{a}}^{(1)}$ word bereken deur Matlab se 'lpc'-funksie, vanaf 'n vektor wat soos volg lyk:

$$q_i = \begin{cases} s_i\ , & i \in W \setminus V \\ \hat{x}_i\ , & i = V \end{cases}\ .$$

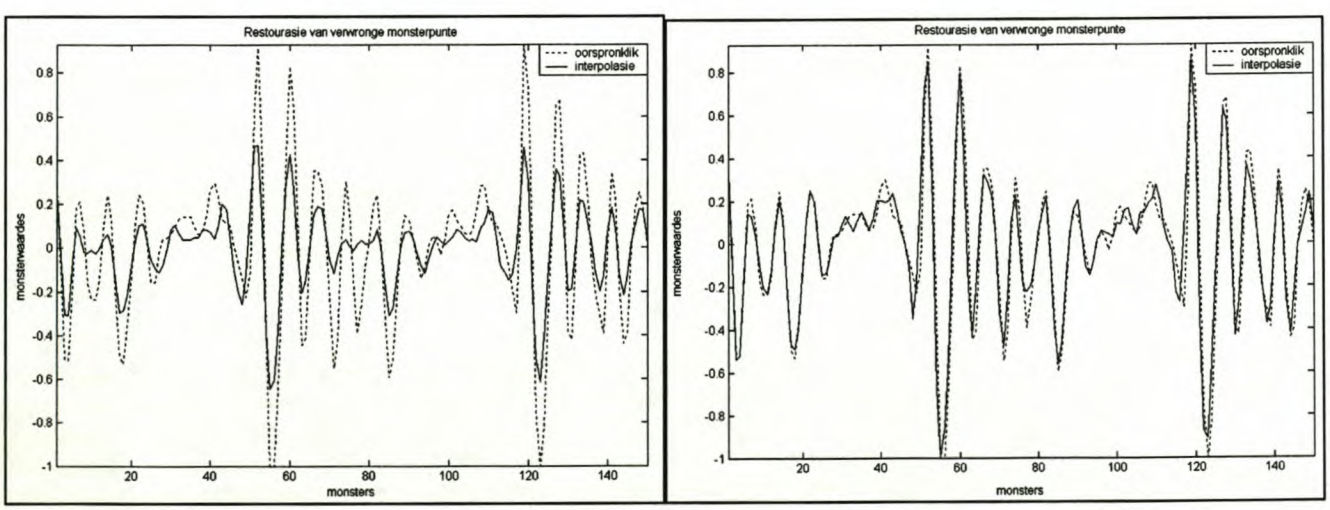


Figuur 41 – Blokdiagram wat die laevlakwerking van die oudiorestorasieprogram toon.

$\hat{\mathbf{x}}^{(1)}$ kan nou bereken word vanuit die afskatting van $\hat{\mathbf{a}}^{(1)}$. Hierdie oplossing van die onbekende monsters word nou weer gebruik om nuwe AR-parameters $\hat{\mathbf{a}}^{(2)}$ van 'n nuwe q_i af te skat, waarvan 'n nuwe $\hat{\mathbf{x}}^{(2)}$ weer afgeskat kan word. Hierdie iteratiewe proses word herhaal totdat 'n bevredigende resultaat verkry is. Die algoritme konvergeer baie vinnig; tipies na drie tot vier iterasies.

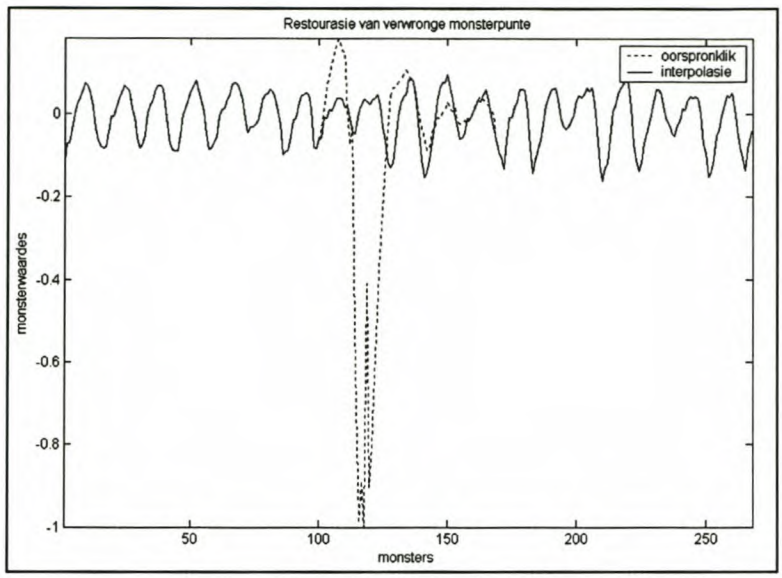
7.1.3 Resultate

Tydens restourasie is die orde van die AR-afskatting as $p \geq 150$, afhangend van die interpolasie resultaat, gekies. 'n Orde van $p = 150$ lewer genoeg detail om spraak baie goed te modelleer, maar indien 'n gaping van korrupte monsters heelwat groter as $m = 150$ is, moet die orde van die AR-afskatting ook verhoog word.



(a) **(b)**

Figuur 42 - Interpolasie van verwronge monsters na (a) een iterasie en (b) tien iterasies.



Figuur 43 - Restourasie van verwronge monsters na tien iterasies.

Resultate van die algoritme word in Figuur 42 en Figuur 43 getoon. Figuur 42(a) en Figuur 42(b) toon resultate van bekende toetsdata van 150 datapunte van 'n onverwronge stuk spraak. Die orde van die AR-afskatting was $p = 150$ en is van 300 korrekte datapunte (150 aan weerskante van die verwronge seleksie) afgeskat. Die gebroke lyn is die oorspronklike stuk spraak en die soliede lyn is die interpolasie wat deur toepassing van die algoritme verkry is. Figuur 42(a) toon die interpolasie na een iterasie en dit is reeds baie duidelik dat die interpolasie die korrekte vorm begin aanneem. Figuur 42(b) is na 10 iterasies en die interpolasie vergelyk baie goed met die oorspronklike data.

Figuur 43 toon 'n seksie rou data afkomstig van 'n wassilinder. Die gebroke lyn toon die oorspronklike verwronge data terwyl die soliede lyn die interpolasie na 10 iterasies is. Die SNR van die klanksnit na oudiorestorasie was 12.29dB.

7.2 Short-time spectral attenuation

STSA ("short-time spectral attenuation") is 'n enkelintree ruisverminderingsmetode wat tydveranderlike verswakking aan die korttydspektrum van die ruiserige sein aanbring, om sodoende die onderliggende oorspronklike sein te verhef. STSA metodes is nie-parametries en van die mees populêre metodes van spraakverheffing en oudiorestourasie.

Die model vir die waargenome sein is 'n bytellende ruis model:

$$y(n) = x(n) + d(n)$$

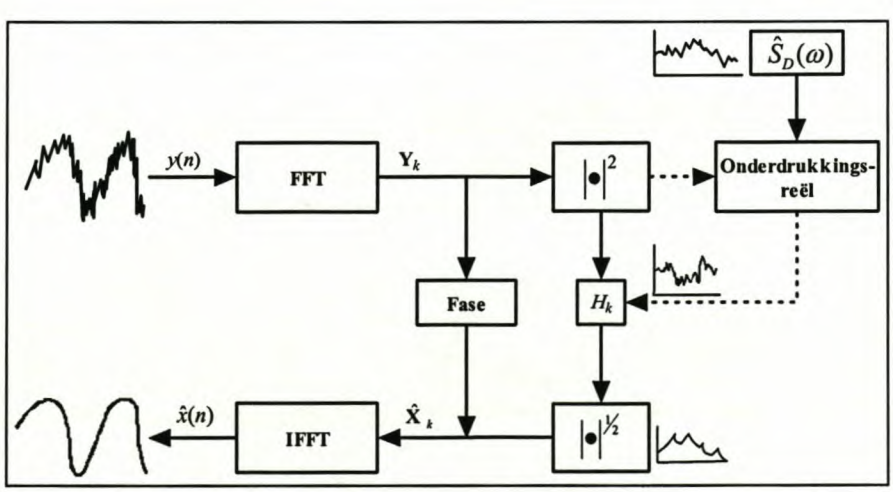
waar $x(n)$ die oorspronklike sein is, $d(n)$ lukrake ruis en $y(n)$ die waargenome sein.

Verwerking van $y(n)$ geskied deur die oorvleuel-en-sommeer metode van korttyd Fourier analise en sintese, waartydens die sein deur die toepassing van die korttyd Fouriertransform (STFT) geanaliseer word, nadat die sein verdeel is in oorvleuelende raampies met die toepassing van 'n vensterfunksie. Na aanpassings aan die korttydspektrum, word die raampies invers getransformeer, gevenster, gesommeer en geskaleer om die gerestoureerde sein te sintetiseer.

Hierdie proses van STSA kan gesien word as 'n reële aanwins H_k wat op elke spektrale komponent k van die korttydspektrum $\mathbf{Y}_k$ van die waargenome sein toegepas word om 'n afskatting $\hat{\mathbf{X}}_k$ van die oorspronklike sein $\mathbf{X}_k$ te verkry:

$$\hat{\mathbf{X}}_k = H_k \cdot \mathbf{Y}_k$$

H_k word deur 'n onderdrukkingsreël beheer wat van die relatiewe sein- en ruisdrywingspektra, $S_X(\omega)$ en $S_D(\omega)$, afhanklik is. Aangesien slegs die ruiserige sein $y(n)$ beskikbaar is, moet 'n afskatting van die ruisdrywingspektrum $\hat{S}_D(\omega)$ vanaf dele in $y(n)$ in die afwesigheid van die onderliggende sein, verkry word, onder die aanname dat die ruis dieselfde vir die seinlose en seinbevattende intervalle is.



Figuur 44 - Blokdiagram wat die werking van STSA toon.

Die belangikste en mees komplekse deel van die STSA algoritme is die onderdrukkingsreël. Die mees suksesvolle onderdrukkingsreël is dié voorgestel deur Ephraim en Malah [15], [16]. Hierdie onderdrukkingsreël verminder die musikale ruistone wat tipies deur STSA veroorsaak kan word. Wolfe [17] het hierdie onderdrukkingsreël verder aangepas om die waarnemingseienskappe van menslike gehoor in ag te neem om die musikale klanke verder te elimineer.

'n Gevorderde STSA algoritmestel is deur Wolfe vir Matlab geskryf. Dit is gebruik om die klanksnit te restoureer. Parameters wat die onderdrukkingsreël beïnvloed is die ruisvloer ("noise floor" of "residual noise"), ruisdrywingspektrum en alpha-parameter van die Ephraim en Malah reël wat vir alle praktiese doeleindes op $\alpha = 0.98$ vasgestel is [16].

Die doel van die ruisvloer-parameter is bloot om 'n mate van outentisiteit van die oorspronklike opname behoue te laat bly. Die ruisdrywingspektrum word verkry deur 'n periodogramafskatting vanaf seinlose segmente van die waargenome sein $y(n)$.

Ander parameters wat verstel kan word is die raamlengte, persentasie oorvleueling tussen raampies en vensterfunksie. Die raamlengte is 'n belangrike keuse, aangesien 'n raampie wat te kort is, te min detail aan die korttydspektrum toeken. Elke STFT-komponent dek dan 'n groot bandwydte. Dit veroorsaak dat, buiten die ruis, die spraak ook beduidend deur die onderdrukkingsreël beïnvloed kan word. 'n Raamlengte wat weer te groot is, kan smeereffekte in die verwerkte sein tot gevolg hê. Die raamlengte is gekies as 1024 monsters wat beteken dat elke raampie $\frac{\text{aantal monsters}}{F_s} = \frac{1024}{12500} = 81.92\text{ms}$ lank is. Dit lewer genoeg resolusie in die frekwensiegebied om genoegsame aanpassings aan die spektrum aan te bring, en is ook kort genoeg om smeereffekte in die tydgebied te voorkom.

7.2.1 Resultate

Die resultate van die STSA-metode word saam met dié van die LPC-filter spraakverheffings-metode aan die einde van hierdie hoofstuk behandel.

7.3 LPC-spraakverheffing

Hierdie metode is die eenvoudigste en het tot op hede van die beste resultate gelewer. Die eenvoud van die toepassing van die metode maak dit baie wenslik.

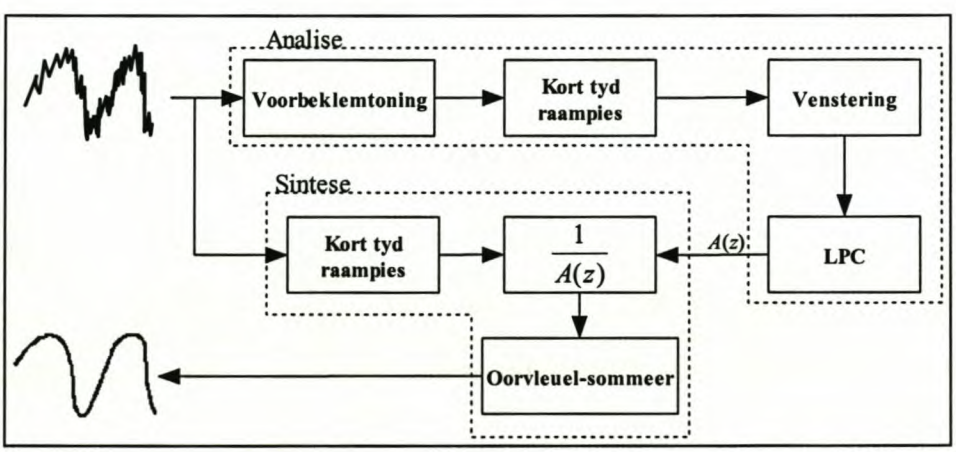
7.3.1 Toepassing van die metode

Die metode is in Matlab toegepas. 'n Voorbeklemtoningsfilter ('pre-emphasis filter') word eers op die klanksnit toegepas. Die funksie van die voorbeklemtoningsfilter is om die hoër frekwensiekomponente van die spraak te verhef. Laer formante wat stemhebbende klanke verteenwoordig, dra in spraakseine meer energie as die hoër formante. Die energie van die sein word meer egalig in die spektrum versprei deur voorbeklemtoning toe te pas. Hoër frekwensie klanke, wat frikatiewe klanke insluit, word nou beter in die afgeskatte LPC-modelparameters ingesluit. Die voorbeklemtoningsfilter is bloot 'n eenvoudige hoogdeurlaatfilter met die volgende oordragfunksie:

$$H(z) = 1 - 0.97z^{-1}$$

Die LPC-parameters word van hierdie gefilterde klanksnit afgeskat.

Die klanksnit word geanaliseer met die oorvleuel-en-sommeer-metode, met die toepassing van 'n vensterfunksie, met 50% oorvleueling tussen raampies. Die doel van die vensterfunksie is om die diskontinuïteit by die rande van die raampie te verminder, wat foute tydens die LPC-afskatting kan veroorsaak. Enige van 'n reeks vensterfunksies wat dalend na die rande van die raampie is, kan gebruik word. In hierdie toepassing is die Hamming-venster gekies aangesien dit 'n algemene venster



Figuur 45 - Blokdiagram wat die werking van LPC-spraakverheffing toon.

met 'n gladde vorm is.

Die orde van die LPC-afskatting is 'n belangrike besluit. 'n Te lae orde kan te min detail aan die spektrum toeken, terwyl 'n te hoë orde skerp valse pieke in die spektrum tot gevolg het. LPC-ordes tussen 64 en 128 het nie beduidende hoorbare verskille in die verhefte klanksein tot gevolg gehad nie. Daar is besluit op 'n orde van 128 omdat 'n hoër detail spektrum meer detail van die spraak behoue sal laat bly.

Die LPC-parameters vir elke korttydraampie word afgeskat deur van die 'lpc'-funksie in Matlab gebruik te maak. Noudat die LPC-parameters vir elke spesifieke tydraampie vanaf die voorbeklemtoonde klanksnit bepaal is, word die ooreenstemmende tydraampies van die oorspronklike klanksnit met die LPC-filter gefilter om die spraak te verhef. Daar bestaan egter nog 'n klein probleem wat aangespreek moet word.

Stemhebbende klanke wat feitlik sinusvormig is, lewer LPC-spektra met skerp resonante pieke. Wanneer hierdie klanke met die sterk resonante LPC-filter gefilter word, word hierdie klanke heelwat in vergelyking met ander minder resonante klanke, versterk. Hierdie verskynsel maak die resonante klanke onnatuurlik hard en steurend, en is op twee maniere onderdruk.

Eerstens is die resonante pieke van die LPC-spektra onderdruk deur die afgeskatte LPC-parameters aan te pas deur die pole van die LPC-filter verder van die eenheidsirkel op die z-vlak te plaas. Dit is gedoen deur die pole van die filter met 'n faktor $r<1$ te vermenigvuldig. Dit kan ook gedoen word deur die vektor waarin die parameters gestoor is met 'n funksie te vermenigvuldig. So word die Q-waarde van die resonante pieke ietwat verlaag. Die funksie waarmee die LPC-parameters aangepas is, word in bylae E bespreek.

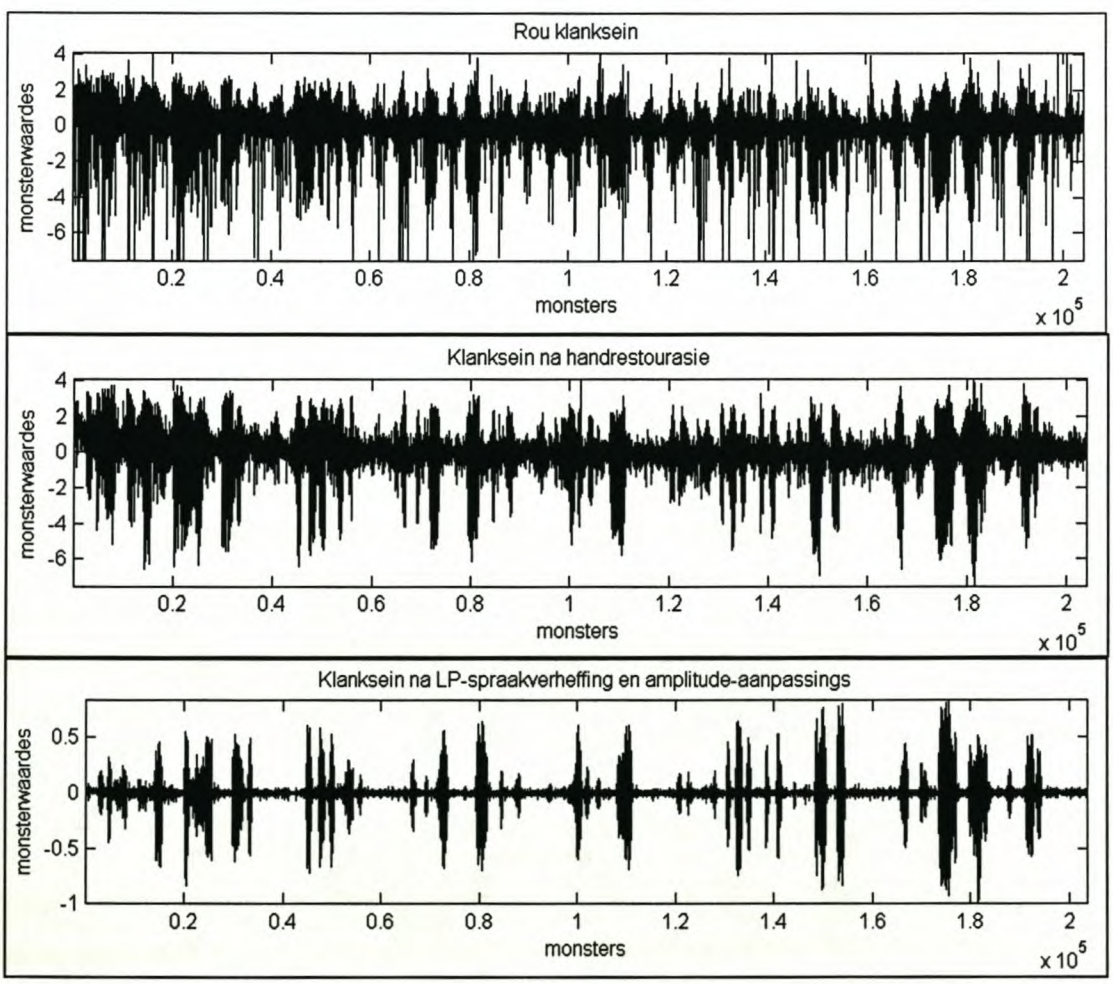
Die tweede aanpassing wat gemaak is om die onnatuurlik harde resonante klanke te onderdruk, is om die baie harde klanksegmente, wat beduidende groot amplitude in die tydgebied het, in die klankverwerkingsprogram, "CoolEdit", te versag om sodoende 'n meer natuurlike weergawe te lewer. Hierdie aanpassing is gedoen deur die "envelope"-funksie op verskeie groot-amplitude segmente in die tydgebied toe te pas, totdat 'n bevredigende resultaat behaal is.

7.3.2 Resultate

Die resultate van LPC-spraakverheffing word saam met STSA se resultate in die volgende afdeling behandel.

7.4 Resultate en Samevatting

Figuur 46 toon die verwerking van die klanksein na die verskillende verwerkingstappe. Luistertoetse is uitgevoer deur 16 luisteraars om die twee spraakverheffingsmetodes te evalueer. Die evaluasie is regverdig en onpartydig uitgevoer deurdat die luisteraars die toets afsonderlik moes uitvoer en nie geweet het na watter klanksnit geluister word nie. Die klanksnitte is op oorfone teruggespeel. Om die evaluasie eenvoudig te hou, is slegs twee vrae per klanksnit gestel:



Figuur 46 – (a) toon die rou klanksein verkry deur dwarsskandering. (b) toon die klanksein na oudiorestourasie. (c) is die klanksein na LP-spraakverheffing.

1. Duidelikheid/Verstaanbaarheid van spraak.
2. Graad van ruissteurnis (Hoe steurend is die ruis?).

Beide die vrae is op 'n skaal van 1 tot 3 geëvalueer, waar 1 "onduidelik / nie steurend" verteenwoordig en 3 "duidelik / erg steurend". Die uitslag van die luistertoetse word in die tabel onder op die bladsy opgesom. Die duidelikheid-ruis-verhouding is bereken deur die duidelikheidstelling deur die ruissteurnistelling te deel. Bevindings na aanleiding van die evaluasie, was soos volg:

Hoewel STSA ernstige ruis feitlik geëlimineer het, is die duidelikheid en verstaanbaarheid van die spraak negatief beïnvloed en klink sommige woorde onduideliker as in die oorspronklike opname. Baie sagte klanke is ook deur STSA verswak in plaas van verhef. Die SNR vir die finale STSA-resultaat is 32.06.dB

Aan die ander kant is gevind dat die LPC-spraakverheffing die duidelikheid en verstaanbaarheid van die spraak behou, terwyl ruisverwerping, teenoor STSA, nie so goed is nie. Dit is verstaanbaar omdat die LPC-metode die spraakmodel verhef en so ook dus die ruis baie gekleurd maak. Die voordeel was wel dat hierdie gekleurde ruis nie te steurend vir die luisteraars voorkom nie, aangesien die spraak die ruis genoegsaam oortref. Die SNR vir die finale LPC-resultaat is 26.08dB.

Uitslag van luistertoetse			
Klanksnit	Duidelikheidstelling	Ruissteurnistelling	Duidelikheid-ruis-verhouding
Rou klanksnit	33	46	0.717
STSA	22	25	0.88
LPC	35	32	1.094

Tabel 3 - Uitslag van die luistertoetse soos geëvalueer deur 16 luisteraars.

Opsomming van SNR van rou klanksnit tot verhewe klanksnit	
SNR van rou klanksnit	2.77dB
SNR na handrestourasie	12.29dB
SNR na handrestourasie en STSA	32.06dB
SNR na handrestourasie en LPC-spraakverheffing	26.08dB

Tabel 4 - Opsomming van die SNR na elke verwerkingstap.

Hoofstuk 8 - Gevolgtrekkings en Aanbevelings

Hierdie hoofstuk is 'n samevatting van die tesis, waar die gevolgtrekking ten opsigte van die metodes wat ondersoek is, bespreek word. Aanbevelings vir moontlike verbetering en toekomstige uitbreiding word ook gemaak.

8.1 Gevolgtrekkings

'n Opneemtoestel is ontwerp en vervaardig. Die meganiese toestel is losstaande van die opneemsensors om proefneming met verskillende sensors te vergemaklik.

Twee opneemmetodes is ondersoek. Die eerste is 'n mikroskopiese beeldmetode waardeur mikroskopiese beelde van die silinderoppervlak geneem word om die wydtemodulasie wat die klankgroefie besit, in klank om te sit. Hoewel resultate behaal is, kan die resultate nie as baie betroubaar of van hoë kwaliteit geag word nie. Dit is toe te skryf aan 'n paar redes. Die resolusie van die optiese stelsel met die webkamera, lewer slegs 7-bis data. Die groefoppervlak word gewoonlik die ergste en maklikste deur krapmerke beskadig, wat beteken dat data afkomstig van die groef se oppervlakterand die swakste moontlike data op die silinder is. Die wydtemodulasie van die groefie is ook nie die ware klankseinmodulasie nie, maar slegs 'n sekondêre verskynsel van die V-vorm van die oorspronklike opneemnaald, aangesien die klanksein eintlik in die diepte van die silinderoppervlak gegraveer is. Weens hierdie redes is besluit om 'n tweede opneemmetode te implementeer wat die dieptemodulasie van die groefie direk meet.

Die tweede opneemmetode wat ondersoek is, was dieptemonstering deur middel van 'n CD-speler se lasersensor. 'n Beperkte begroting het nie toegelaat dat 'n hoë kwaliteit mikrodieptemeter aangekoop kon word nie. 'n Lasersensor van 'n CD-speler is aangepas om as 'n mikrodieptemeter te funksioneer en die dieptemodulasie van die groefie te meet. Die lasersensor is egter nie ideaal nie, aangesien die laserdiode 'n beperkte leeftyd het en, indien dit sou ingee, dit nie maklik bloot met 'n ander vervang kan word nie. 'n Herontwerp sal gedoen moet word en datavelle van sulke lasersensors is nie geredelik beskikbaar nie.

Drie metodes van laserskandering is ondersoek, naamlik voorwaartse opname, truwaartse opname en dwarsskandering. Voorwaartse opname geskied deur monstering van die dieptemodulasie van die groefie soos die groefie teen die korrekte steek in die voorwaartse rigting onder die lasersensor roteer. Truwaartse opname is identies behalwe vir die feit dat die silinder bloot truwaarts onder die lasersensor, teen die korrekte steek, roteer. Beide hierdie opneemmetodes dek een groefbreedte deur vyf opnames wat een vyfde van 'n groefbreedte uitmekaar gespasieer is. Die beste vier van hierdie vyf opnames word gebruik om die gemiddelde opname te bepaal. Sigbare en hoorbare verbeteringe in die SNR kan opgemerk word tussen 'n enkelopname en die gemiddelde opname. Aangesien so 'n verbetering in SNR bereik is deurdat 'n aantal monsterpunte in die groefie bydra tot 'n klankmonster, is besluit om die hele groefprofiel te monster, wat aanleiding gegee het tot die dwarsskanderingmetode.

Dwarsskandering is 'n nie-kousale opneemmetode wat dwars oor die breedte van die heliks van groefies skandeer, terwyl die silinder onder die lasersensor transleer. 'n Hele rotasie word opgebou deur 5000 dwarslyne onder mekaar te stapel. 'n Volledige beeld van die silinderoppervlak word so gevorm. Tydens die verwerking van die beeld tot klank, word altesaam 80 monsterpunte vanuit die groefie se profiel verenig om 'n klankseinmonster te lewer. Dit het beduidende verbeteringe in die SNR tot gevolg.

Klank is verkry, hoewel die kwaliteit van die spraak ongelukkig nie heeltemal na wense is nie. Die probleem is ongelukkig nie so eenvoudig nie, aangesien daar nie 'n absolute generiese oplossing vir die klankonttrekking is nie. Elke silinder is uniek en het sy eie struikelblokke wat oorkom moet word. Dit maak die hele proses van klankonttrekking en -verwerking baie tydrowend en arbeidsintensief.

Een van die silinders wat ondersoek is, is gelas en het onegalig geroteer. Die kraaklaste was grotendeels aksiaal gerig en kon verwyder word deur die metodes wat in afdeling 6.3 bespreek is. Krake wat wel diagonaal of saam met die groefrigting loop, is nie in hierdie tesis ondersoek nie.

'n Tweede silinder het 'n strook wat baie ernstig deur kepe en krapmerke beskadig is, besit. Dit was moontlik deur hardhandige hantering of verkeerde storing veroorsaak.

Die nagevolge van hierdie swak klankstrook is in die tydgebied digitaal gerestoureer deur die ergste verwronge seksies met interpolasies op te vul. Om betroubare data van elke silinder te verkry, moet hierdie unieke probleme van elke silinder ondersoek en opgelos word, wat 'n tydrowende proses is.

Die tipe ruis teenwoordig in die rou klanksein maak restourasie en spraakverheffing moeilik. Die produk van 'n keep of kraak is nie bytellende gaussiese witruis nie, maar eintlik 'n gaping van vermiste data. Hierdie gapings is moeiliker om te identifiseer nadat die klanksein reeds gefilter is. Die kepe is makliker op die dwarsskanderingdata sigbaar, aangesien die data nog nie gefilter is nie. Die rou dwarsskanderingdata is dus makliker om met die hand te restoureer as die voorwaartse opname se data, wat reeds sterk gefilter is deur die bandwydte van die lasersensor se terugvoerstelsel.

'n Handoudiorestourasieprogram is suksesvol in Matlab geïmplementeer. Die program skat hoë kwaliteit interpolasies vir verwronge en vermiste datasegmente af deur wegings vanuit die naasliggende korrekte datapunte. Die program is ook nuttig vir gebruik met seine anders as klank of spraak, wat voldoende as 'n outoregresiewe proses voorgestel kan word. Die klanksein vanaf die dwarsskandering metode is die rouste moontlike data wat van die silinder onttrek is, daarom is die produkte van kepe en krapmerke baie duidelik sigbaar in die data. Die feit dat die kepe en krapmerke so duidelik sigbaar is, maak hierdie verwronge segmente maklik identifiseerbaar en kan eenvoudig met beter interpolasies vervang word. Na oudiorestorasie was daar 'n 9.5dB verbetering in die SNR.

Twee metodes vir spraakverheffing was ondersoek. Die eerste was STSA ("short-time spectral attenuation"). Die tweede was 'n lineêre voorspellingskoëffisiëntmetode. Hoewel STSA een van die mees algemeen gebruikte metodes vir oudiorestourasie is, het dit nie bevredigende resultate gelewer nie. Witter tipe ruis word baie goed onderdruk, maar dit vaar nie so goed met gekleurde ruis nie. Waar ruis wel goed onderdruk word, word die duidelikheid en verstaanbaarheid van die spraak negatief beïnvloed. Dit kan toegeskryf word aan die feit dat spraakkomponente beïnvloed word as gevolg van die onderdrukkingsreël. Nie slegs ruis word deur die algoritme onderdruk nie, maar spraak ook. Die spraak word as sulks nie duideliker uitgelig nie, hoewel ruis goed verwerp word.

Die lineêre voorspellingskoëffisiëntmetode het, hoewel dit ruis nie so goed verwerp nie, tog die verstaanbaarheid en duidelikheid van die klank behoue laat bly. Evaluasie van die twee spraakverheffingsmetodes is gedoen deur luistertoetse. Dit het getoon dat die lineêre voorspellingsmetode bo STSA verkies word, aangesien die resultaat van meer verstaanbare gehalte was, terwyl die ruis nie te steurend was nie. Die SNR vir die finale resultaat was 26.08dB, wat 'n 23.3dB verbetering in SNR is in vergelyking met die rou dwarsskanderingklanksein.

Hoewel klank verkry is, is dit moeilik om die sukses van die projek te evalueer, aangesien daar geen verwysing is waarteen die resultate gemeet kan word nie. Die LP-spraakverheffingsmetode is ook nie ideaal nie, aangesien dit die verheffing van klanke soos vokale onregverdig bo meer ruiserige klanke soos frikatiewe bevoordeel. Daar is ook geen ruismodel by die verheffingsmetode ingesluit nie. Die SNR waardes moet net as 'n riglyn beskou word, aangesien dit natuurlik belangriker is dat die verstaanbaarheid van die spraak behou bly of verhef word. Luistertoetse is dus die enigste ware evaluasie van die kwaliteit van die resultate.

8.2 Aanbevelings

Met die huidige aandrywingselektronika kan die rotasiemotor nie die oorspronklike opneemspoed haal nie. 'n Bipolêre PWM-aandrywingskema behoort die stappermotor se spoedbereik te verhoog tot so hoog as die oorspronklike opneemspoed. Hoewel dit moontlik is om die rotasiespoed van die stappermotors te verhoog, is dit tans nie prakties nie, as gevolg van die bandwydtebeperking van die lasersensor. Die huidige sensor vereis dat die opname teen 240 maal stadiger as die oorspronklike opneemspoed moet geskied.

Die lasersensor het slegs 'n beperkte dinamiese lineêre bereik, wat beteken dat indien die modulasie-uitwyking vanaf die silinder groter as die dinamiese lineêre bereik van die sensor strek, die gemete sein dus terugvou. Dit is ongelukkig 'n inherente eienskap van die sensor en kan nie bloot net gekorrigeer word nie. 'n Sensor met wyer dinamiese lineêre bereik sal hierdie probleem dus uitskakel.

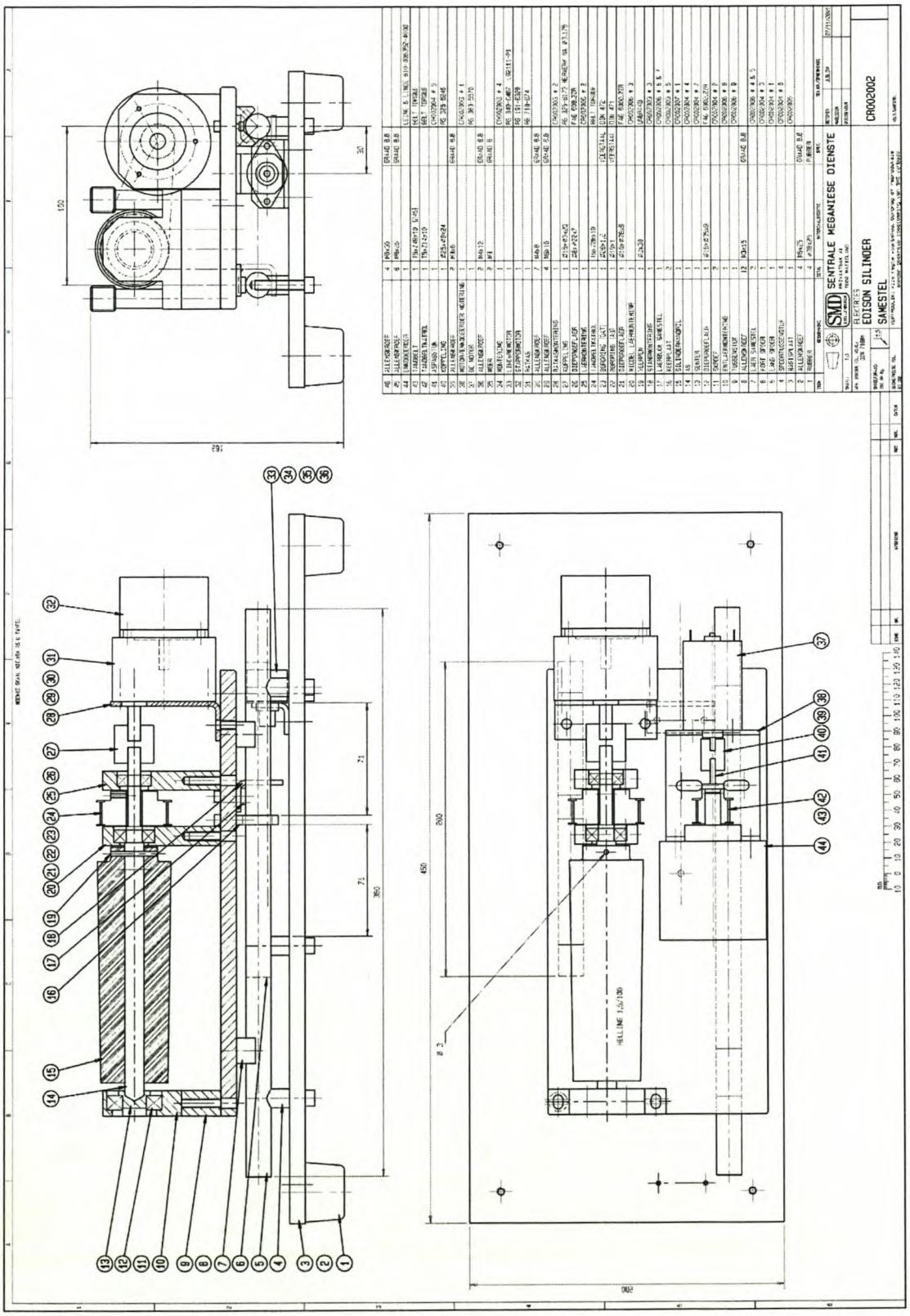
In die huidige eksperimentele opstelling kan die opneemtoestel maklik deur eksterne steurnisse beïnvloed word. Die lasersensor is baie sensitief en voetstappe in die

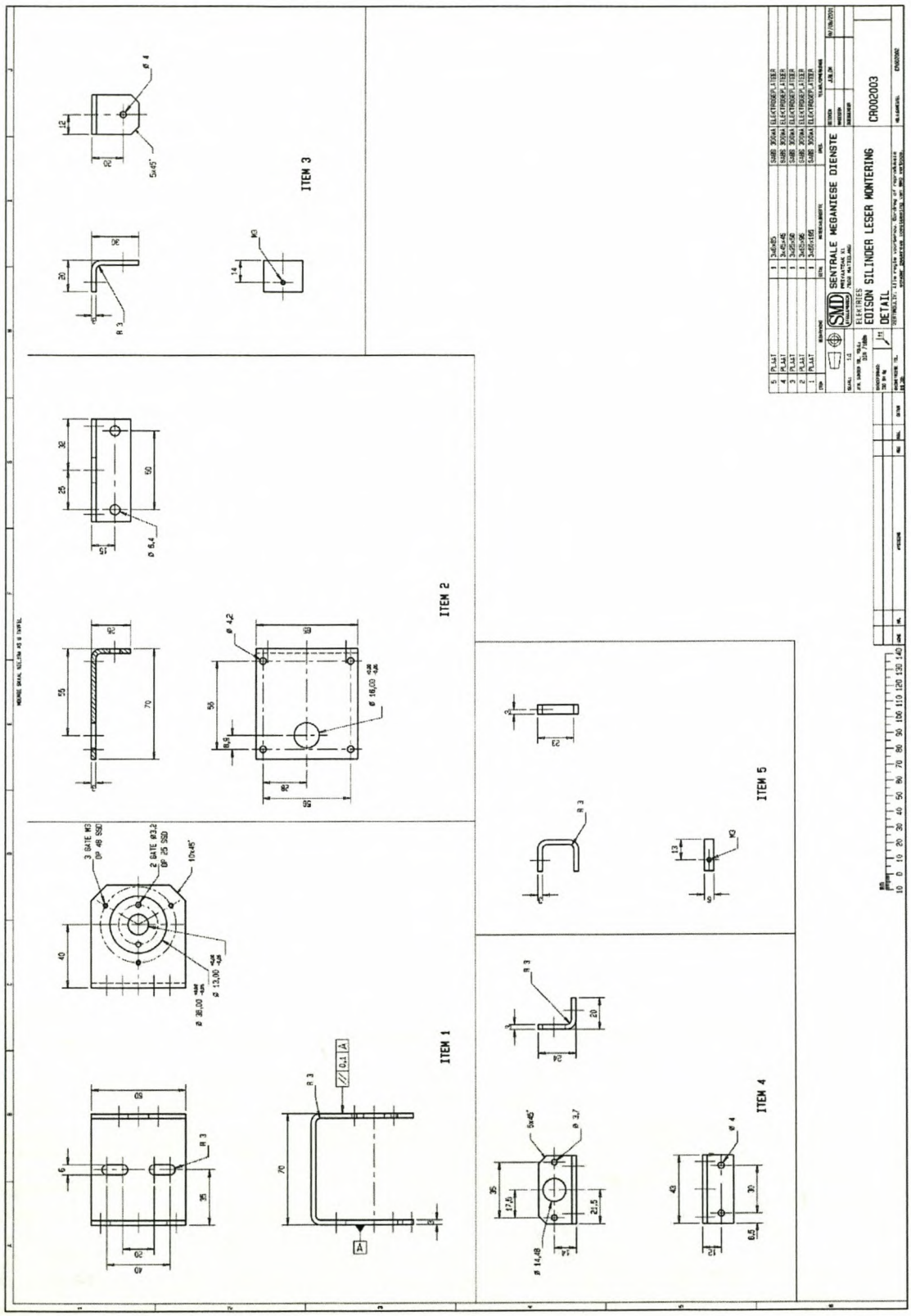
laboratorium waar die opstelling is, kan deur die sensor gemonitor word. Die ideaal sou wees om 'n seismies geïsoleerde opstelling soos dié van Petrov [6] te hê. Dit is egter baie duur.

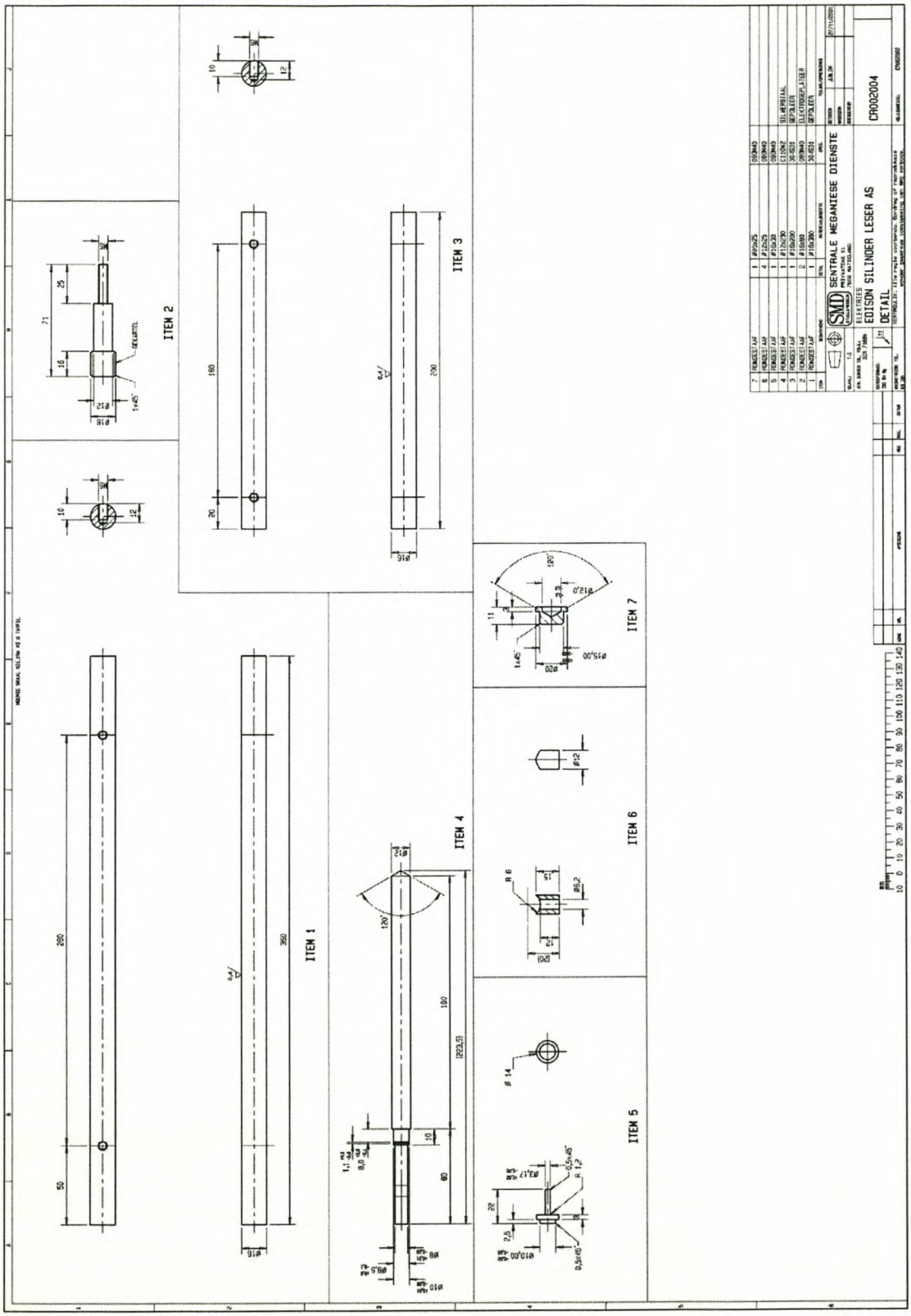
Aangesien 'n LPC-spraakverheffing bo STSA verkies is, kan verwag word dat 'n Wiener, Kalman of 'n ander verwante metode moontlik nog beter resultate vir spraakverheffing kan oplewer. 'n AR-plus-ruis-model wat in [18] bespreek word, behoort 'n eenvoudige implementering te wees met resultate vergelykbaar met wat in hierdie studie verkry is.

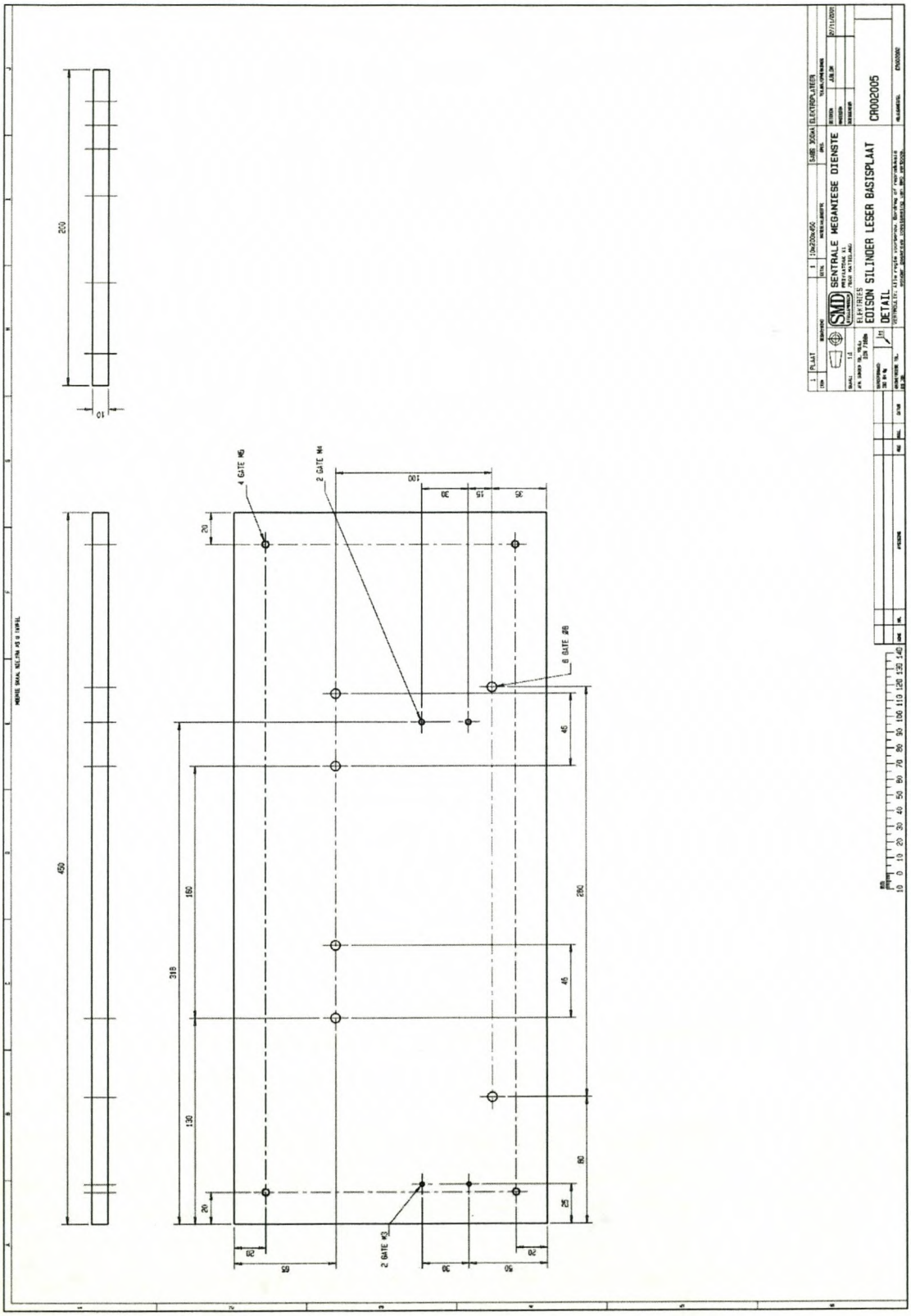
Na veelvuldige luistering van onduidelike spraaksegmente, kan die spraak later wel getranskribeer word. Die feit dat daar juis geweet word wat gesê is, is ook inligting wat gebruik kan word om die ruiserige spraak te verhef. Spraakverheffing kan moontlik gedoen word deur die klanksein in korttydraampies van foneemsegmente te verdeel en elke korttydraampie met die foneemverheffingsfilter te filter. Hierdie foneemfilters sal egter 'n uiters algemene vorm moet hê om te voorkom dat 'n vreemde stem nie op die spreker afgedwing word nie. Om die spraak te transkribeer is ook 'n baie tydrowende en arbeidsintensiewe proses.

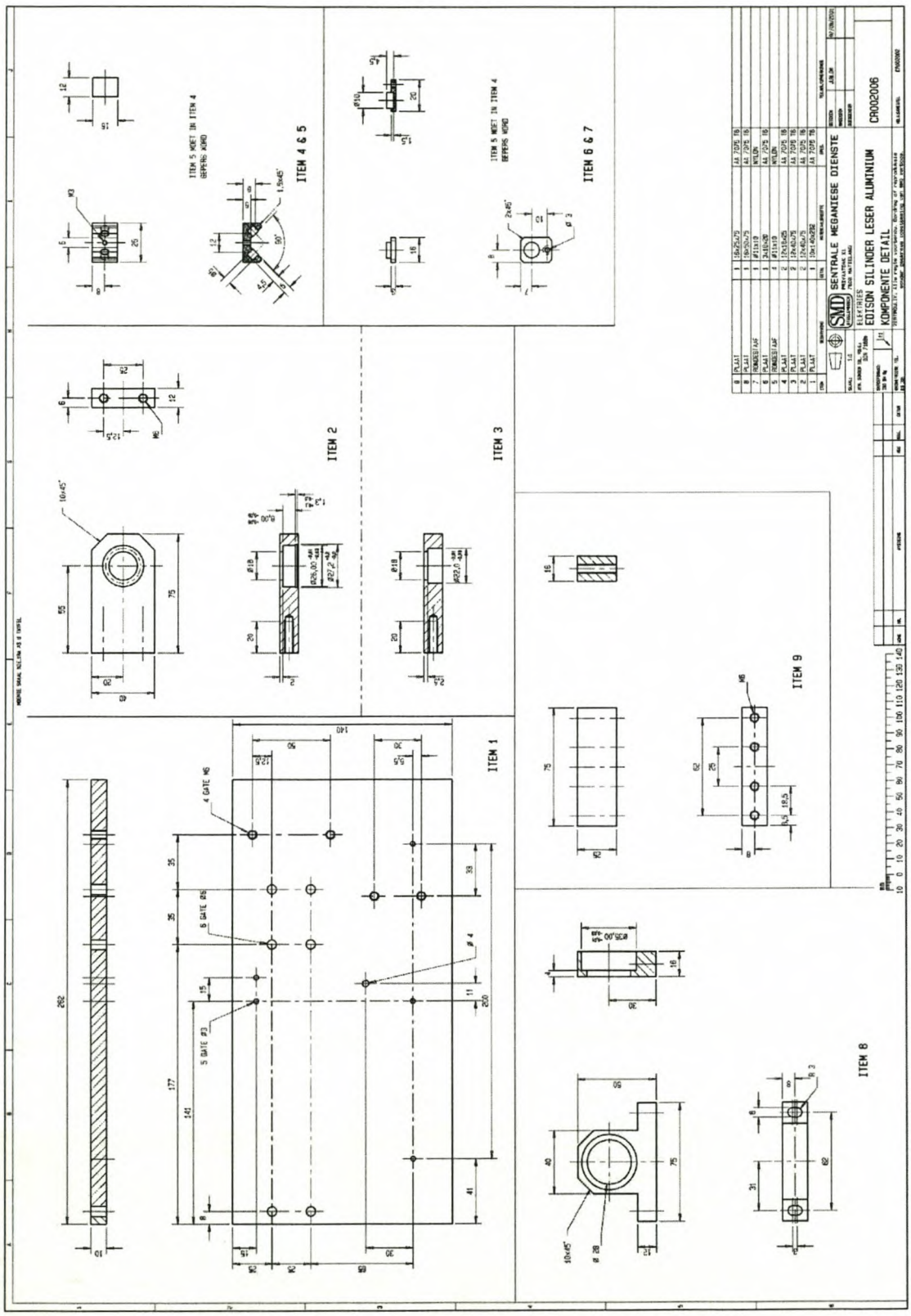
Bylae A – Meganiese tekeninge van die opneemtoestel

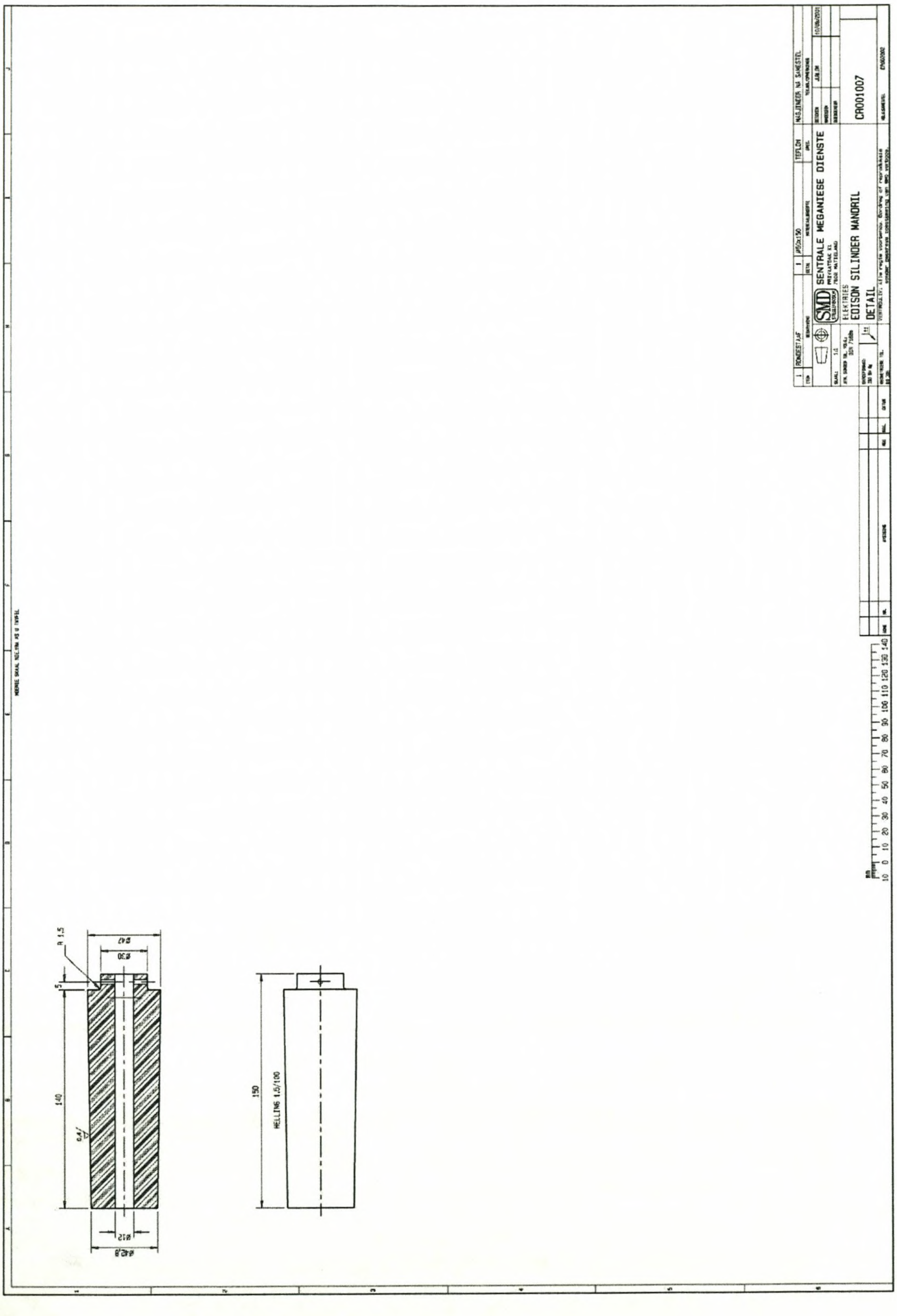




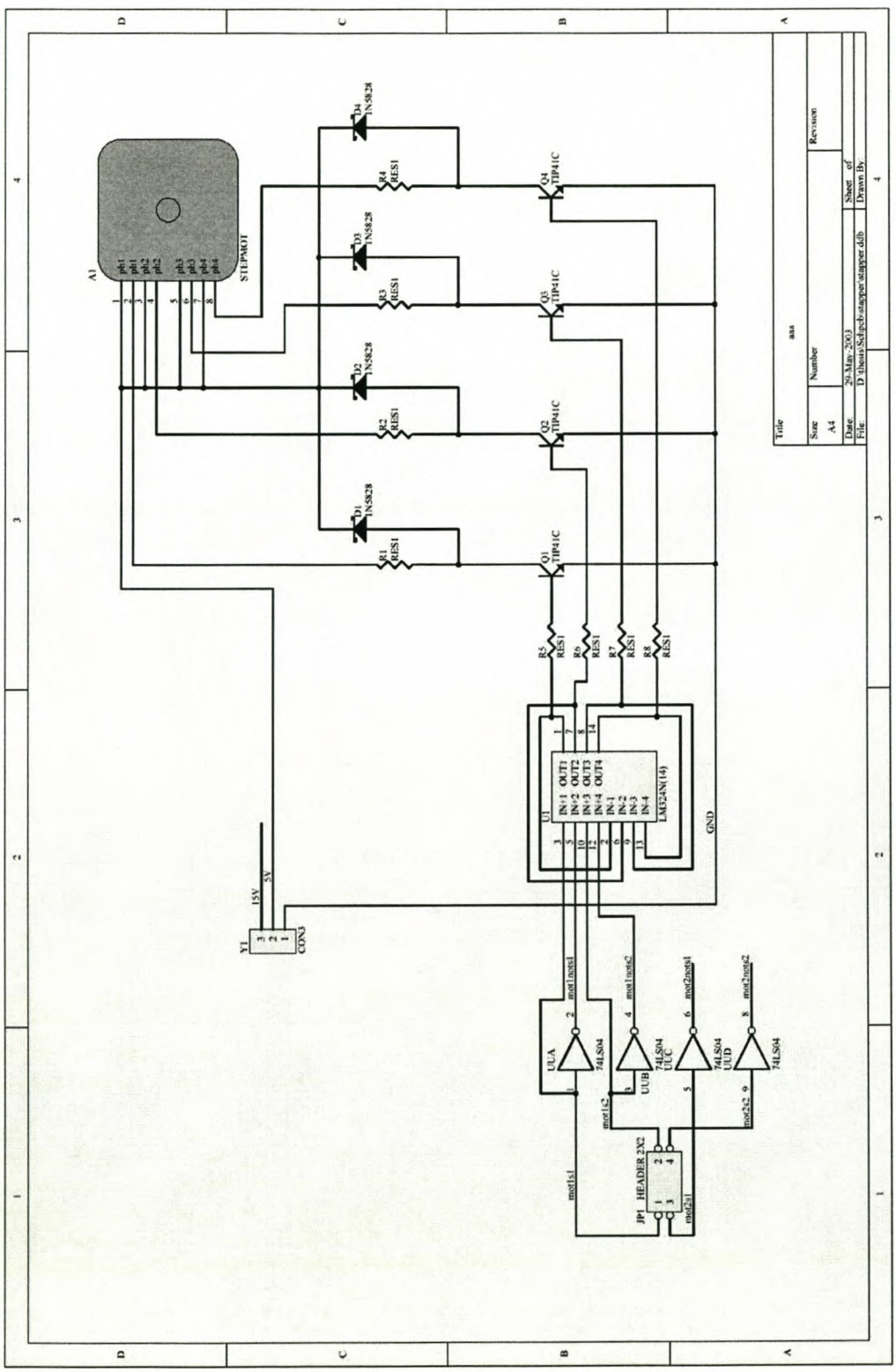




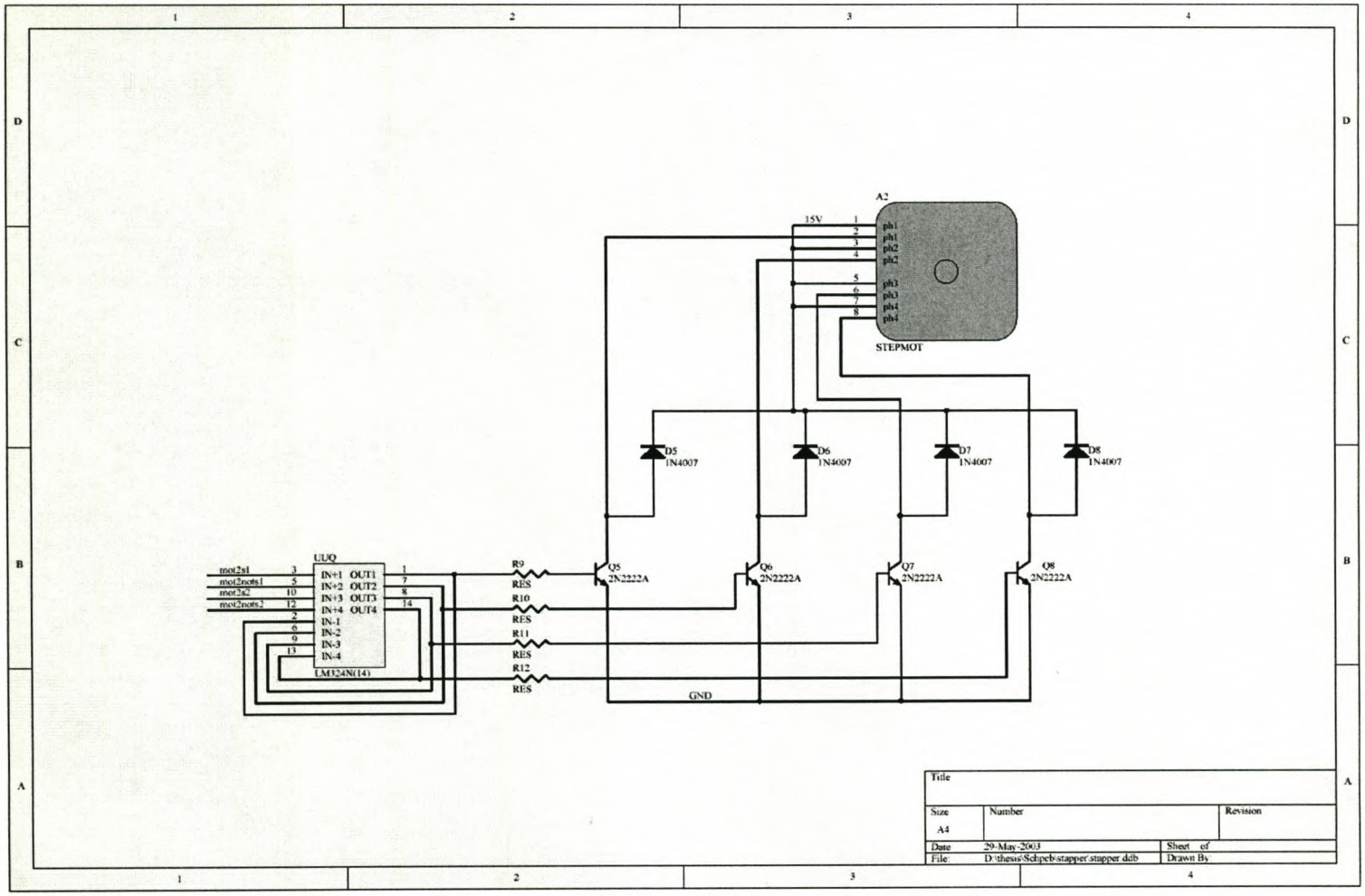

ITEM 1
ITEM 2
ITEM 3
ITEM 4 6 5
ITEM 6 6 7
ITEM 8
ITEM 9
ITEM 5 MOET IN ITEM 4 GEPERS WORD
ITEM 5 MOET IN ITEM 4 GEPERS WORD
4 GATE M6
6 GATE Ø6
5 GATE Ø3
SENTRALE MEGANIESE DIENSTE
EDISON SILINDER LESER ALUMINIUM
KOMPONENTE DETAIL
CR002006


SENTRALE MEGANIESE DIENSTE
ELEKTRIES
EDISON SILINDER MANDRIL
DETAIL
CR001007
TEFLON
RONDESTAAF
HELLING 1,5/100
150
140
R 1.5
Ø42
Ø30
Ø12

Bylae B – Skematiese diagramme van die stappermotor-aandrywing

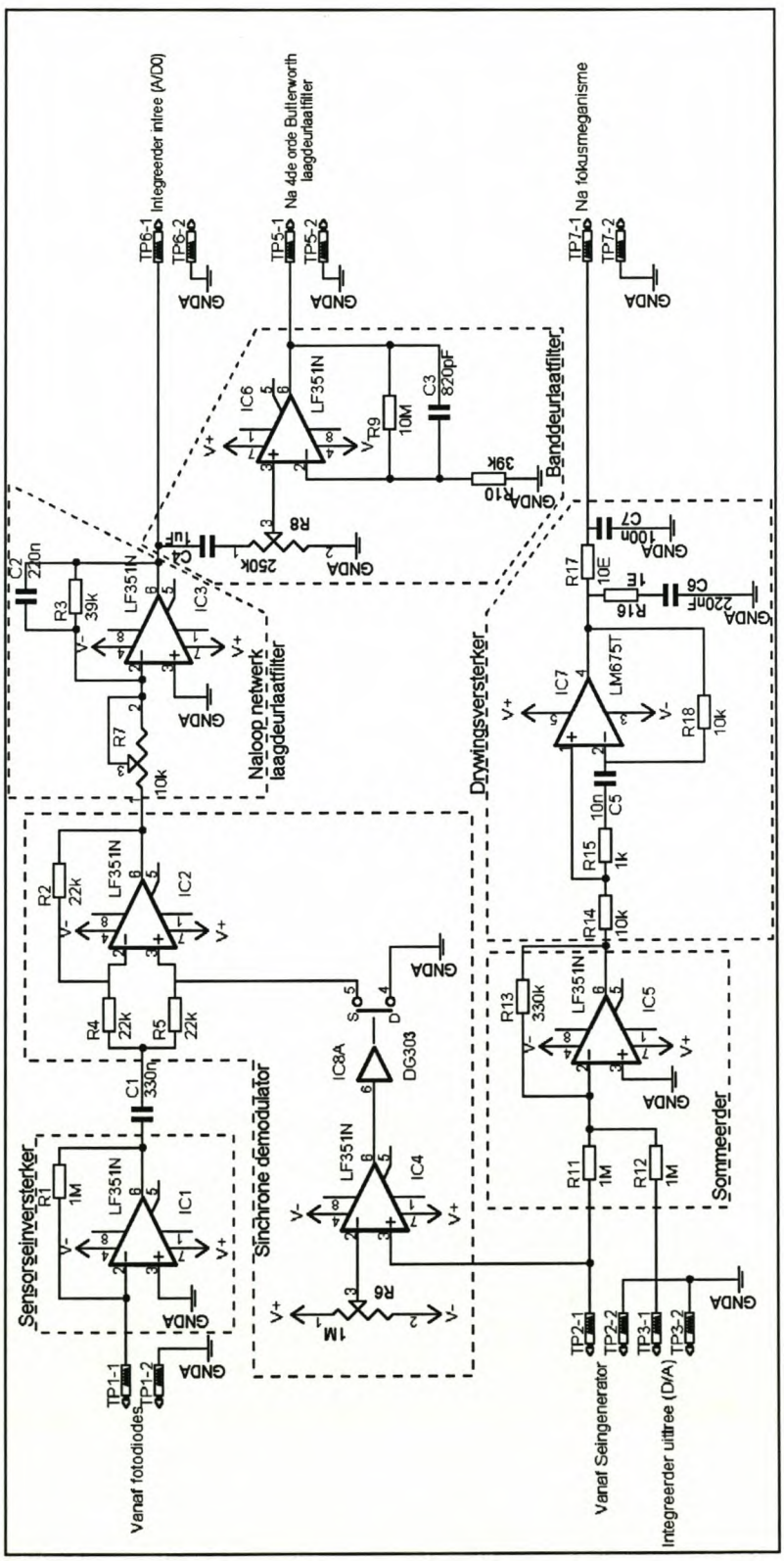


Figuur 47 - Skematiese stroombaandiagram van die rotasiemotoraandrywing.

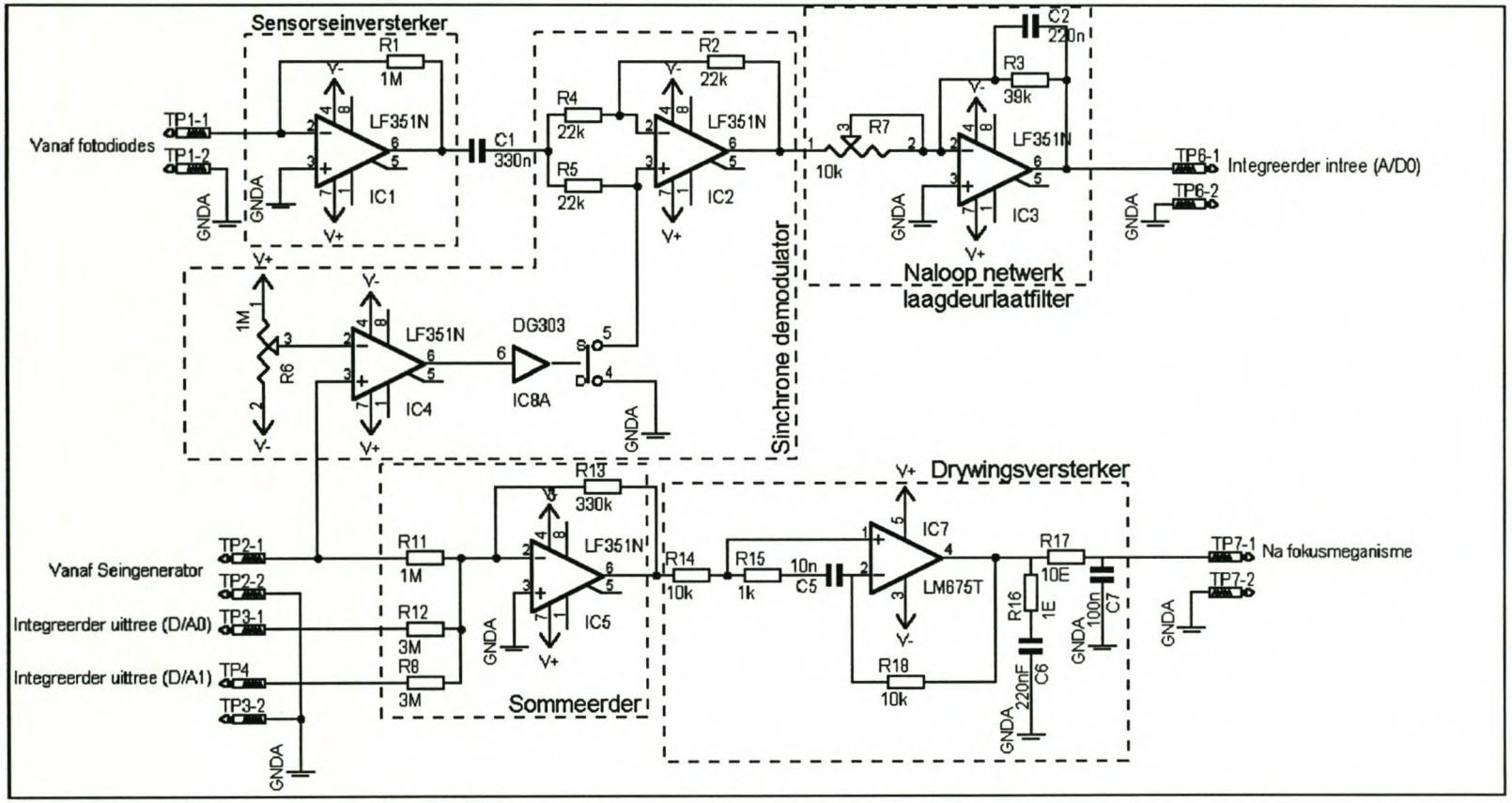


Figuur 48 - Skematiese stroombaandiagram van die translasiemotoraandrywing.

Bylae C – Skematiese diagramme van die laserterugvoerstelsel

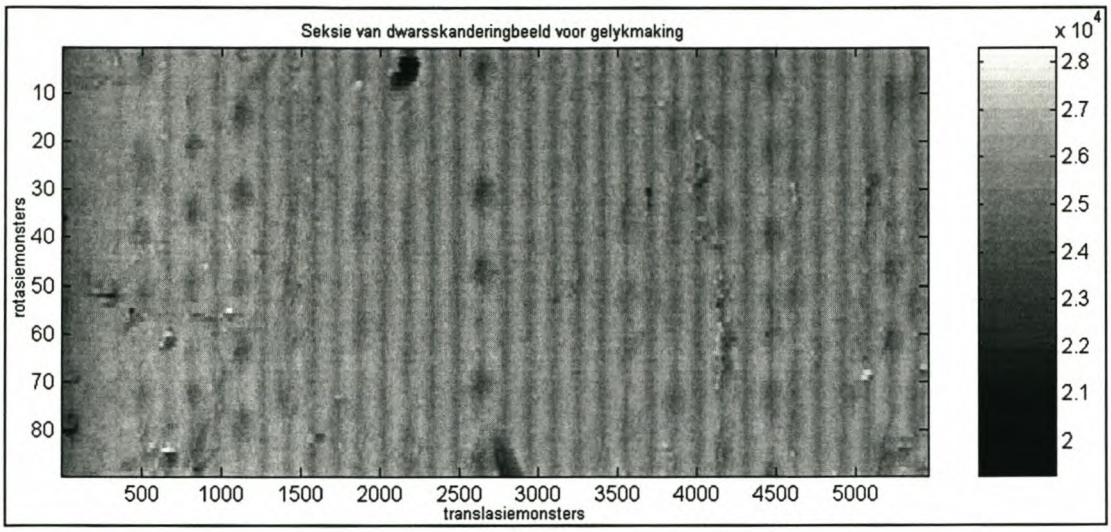


Figuur 49 - Skematiese stroombaandiagram van die lasersensor-terugvoerstelsel.

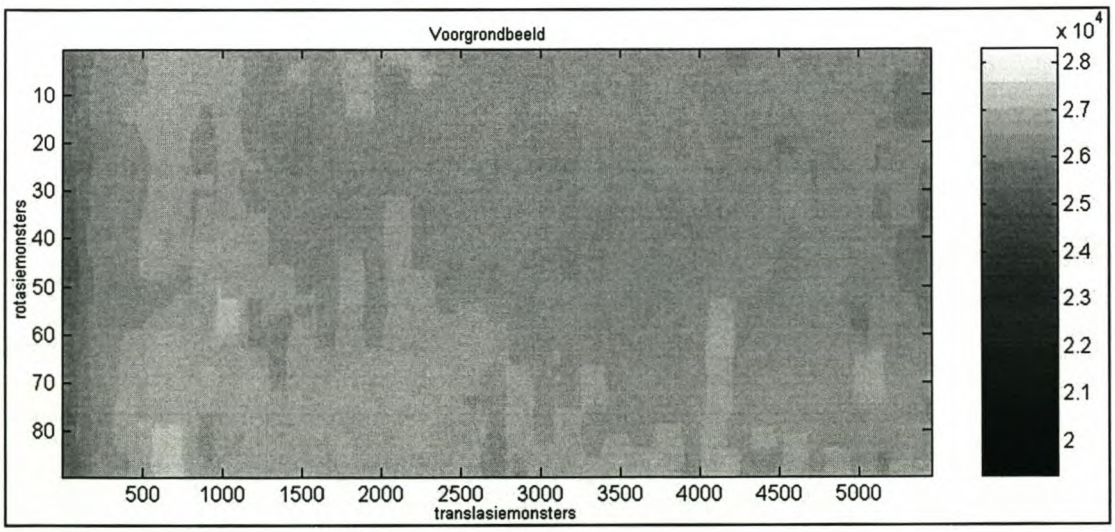


Figuur 50 - Skematiese stroombaandiagram lasersensor-terugvoerstelsel van dwarsskandering.

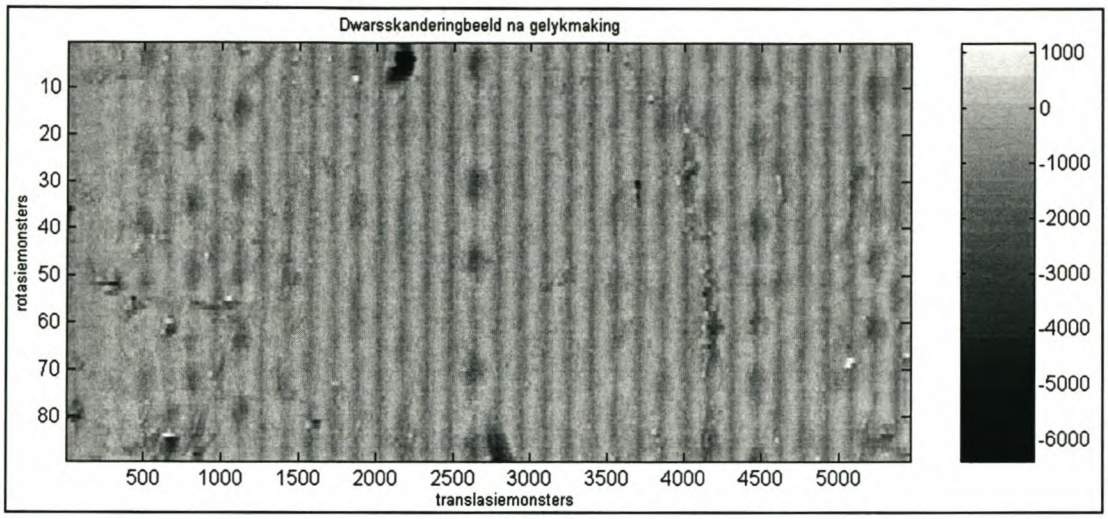
Bylae D - Voorgrondgelykmaking



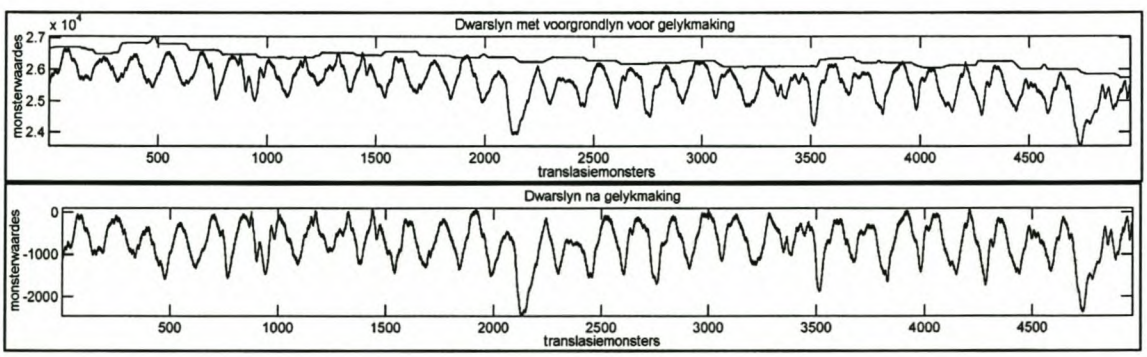
Figuur 51 - Dwarsskanderingbeeld voor gelykmaking.



Figuur 52 - Voorgrondbeeld



Figuur 53 - Dwarsskanderingbeeld na gelykmaking.



Figuur 54 - Dwarslyn met voorgrondlyn, voor en na gelykmaking.

Bylae E - Verband tussen poolaanpassingsfunksie en LPC-filterparameters

Aanvaar $\alpha \pm j\beta$ is 'n komplekse poolpaar net binne die eenheidsirkel, tewyl $a_k, k = 1,2$ die poolpaar se ooreenstemmende karakteristieke polinoomparameters verteenwoordig. Hierdie poolpaar word met 'n faktor $r < 1$ vermenigvuldig om die pole verder van die eenheidsirkel te plaas. Laat $x = r\alpha$ en $y = r\beta$ die nuwe posisies van die poolpaar verder van die eenheidsirkel voorstel, terwyl $b_k, k = 1,2$ die ooreenstemmende nuwe karakteristieke polinoomparameters van die nuwe poolpaar voorstel. Die verband tussen die ou en nuwe pool posisies en parameters is:

$$\begin{aligned}\frac{1}{(z-(\alpha+j\beta))(z-(\alpha-j\beta))} &= \frac{1}{z^2-2\alpha z+(\alpha^2+\beta^2)} \\ &= \frac{1}{1-2\alpha z^{-1}+(\alpha^2+\beta^2)z^{-2}} \\ &= \frac{1}{1+a_1 z^{-1}+a_2 z^{-2}} \\ &= \frac{1}{1+\sum_{k=1}^{2} a_k z^{-k}}\end{aligned}$$

waar $a_1 = -2\alpha$ en $a_2 = \alpha^2 + \beta^2$.

Vervang $\alpha = \frac{x}{r}$ en $\beta = \frac{y}{r}$ in:

$$a_1 = -\tfrac{2x}{r} = \tfrac{1}{r} b_1$$

$$a_2 = (\tfrac{1}{r})^2 (x^2 + y^2) = (\tfrac{1}{r})^2 b_2$$

Die volgende verband is nou dus sigbaar:

$$1 + \sum_{k=1}^{2} a_k z^{-k} = 1 + \sum_{k=1}^{2} r^{-k} b_k z^{-k}$$

$$1 + \sum_{k=1}^{2} r^k a_k z^{-k} = 1 + \sum_{k=1}^{2} b_k z^{-k}$$

Hierdie verband kan uitgebrei word vir enige aantal pole p.

Elke parameter a_k van die filter moet dus met r^k vir $k = 1,2,3...p$ vermenigvuldig word om die pole r maal van die eenheidsirkel te plaas.

Bylae F – Programkode van die stappermotoraandrywing

```
/*;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       Stappermotor pulssein generator n.a.v. 'n pulssein ontvang vanaf
;               die PC na die eksterne onderbrekingspenne van die ATMEL
;
;

;===== Poortpen uitleg =====
;       P1.0 vir rotasiemotorfases 1 en 2
;       P1.1 vir rotasiemotorfases 3 en 4
;       P1.2 vir translasiemotorfases 1 en 2
;       P1.3 vir translasiemotorfases 3 en 4
;       P1.7 vir debug doeleindes vir UART int
;
;       P3.2 en P3.3 is eksterne onderbrekings vir pulse vanaf die PC om die motors te
laat stap
;       P3.4 is vir voorwaartse of truwaartse staprigting vir rotasiemotor en is dus
intreepen vanaf die PC
;       P3.5 is  vir voorwaartse of truwaartse staprigting vir translasiemotor

;===== Serie Opdragte totdusver =====
;       00h - Resetsys
;       01h - Rot.motor GO FWD
;       02h - Rot.motor GO BWD
;       03h - Rot.motor STOP
;       04h - Trans.motor GO FWD
;       05h - Trans.motor GO BWD
;       06h - Trans.motor STOP
;       07h - Zero tellers
*/

#include <reg51.h>

bit UARTserviced = 0, beginrotpos = 0, begintranspos = 0;
unsigned char rotfase = 0, transfase = 0;
unsigned short rotteller = 0;
signed long transteller = 0;
unsigned char wagbinne = 6, wagbuite = 100, serialcommand = 0;

// Funksie vir een voorwaartse rotasiestap
void eenrotstapvoort(void) {
        if (rotfase == 3) {             // siklus van 0123012301230... ens
                rotfase = 0; // indien huidiglik op fase 3, is volgende fase weer 0
        }
        else {
                ++rotfase;              // andersins inkrementeer na volgende fase
        }

        ++rotteller;                    // inkrementeer rotasie posisieteller
        if (rotteller == 5000) {        // indien posisieteller 5000 bereik,
                rotteller = 0;          // herstel teller na 0 (slegs 5000 stappe in 1 rotasie
van silinder)
        }

}

// Funksie vir een truwaartse rotasiestap
void eenrotstaptru(void) {
        if (rotfase == 0) {             // siklus van 32103210321032103... ens
                rotfase = 3; // indien huidige fase = 0, is volgende fase 0
        }
        else {
                --rotfase;              // anders inkrementeer na volgende fase in siklus
        }

        if (rotteller == 0) {           // indien rotasie posisieteller 0 bereik
                rotteller = 5000;       // herstel na 5000 (slegs 5000 stappe in 1 rotasie van
silinder)
        }
```

```
	--rotteller;			// dekrementeer rotasie posisieteller
}

// Funksie vir een voorwaartse translasiestap
void eentransstapvoort(void) {
	if (transfase == 3) {	// werking indenties soos beskryf in funksie vir
rotasiestappe
		transfase = 0;
	}
	else {
		++transfase;
	}

	++transteller;
}

// Funksie vir een truwaartse translasiestap
void eentransstaptru(void) {
	if (transfase == 0) {	// werking indenties soos beskryf in funksie vir
rotasiestappe
		transfase = 3;
	}
	else {
		--transfase;
	}

	--transteller;
}

// Funksie wat rotasiestapsekwensie uitvoer n.a.v.
//	faseposisie verandelike "rotfase"
void toetsrotstap(void) {
	switch (rotfase) {
		case 0:
			P1 |= (1<<1);// P1.1 hoog
		break;
		case 1:
			P1 &= ~(1<<0);		// P1.0 laag
		break;
		case 2:
			P1 &= ~(1<<1);		// P1.1 laag
		break;
		case 3:
			P1 |= (1<<0);// P1.0 hoog
		break;
	}
}

// Funksie wat translasiestapsekwensie uitvoer n.a.v.
//	faseposisie verandelike "transfase"
void toetstransstap(void) {
		switch (transfase) {
			case 0:
				P1 |= (1<<3);// P1.3 hoog
			break;
			case 1:
				P1 &= ~(1<<2);		// P1.2 laag
			break;
			case 2:
				P1 &= ~(1<<3);		// P1.3 laag
			break;
			case 3:
				P1 |= (1<<2);// P1.2 hoog
			break;
		}
}

// Onderbrekingsdiensroetine vir Eksterne onderbreking 0
```

```
//     Hanteer stapinkrement vir rotasiemotor
//     N.a.v. P3.4 se toestant is motor kloksgewys of anti-klok
void ExtInt0(void) interrupt 0 {
       if ( P3 & (1<<4) ) {
              eenrotstapvoort();
       }
       else {
              eenrotstaptru();
       }
}

// Onderbrekingsdiensroetine vir Eksterne onderbreking 1
//     Hanteer stapinkrement vir translasiemotor
//     N.a.v. P3.5 se toestant is motor kloksgewys of anti-klok
void ExtInt(void) interrupt 2 {
       if ( P3 & (1<<5) ) {
              eentransstapvoort();
       }
       else {
              eentransstaptru();
       }
}

// Onderbrekingsdiensroetine vir UART (!Tans in onbruik!)
//     Seriele opdragte word ontvang vanaf die PC vir motor aandrywing
//     Serie opdragte bo-aan file beskryf
void serint(void) interrupt 4 {
       RI = 0;
       UARTserviced = 1;
       serialcommand = SBUF;
}

// 'n wagsiklus vir seker stapspoed (!Tans in onbruik!)
void wagsiklus(void) {
       unsigned int tel;
       for (tel = 270; tel > 1; tel--) {}
}

// Hetstel motors na beginposisies (!Tans in onbruik!)
void resetsys(void) {
       while (rotteller != 0) {
              eenrotstapvoort();
              toetsrotstap();
              wagsiklus();
       }
       while (transteller > 0) {
              eentransstaptru();
              toetstransstap();
              wagsiklus();
       }
       while (transteller < 0) {
              eentransstapvoort();
              toetstransstap();
              wagsiklus();
       }
}

// Hoof programlus
//     Inisialisering en Ondeindige programlus
void main(void) {
       //inisialiseering van poorte
       P1 = 0xff;
       P3 = 0xff;

       // init van UART
       PCON = 0x00;          //double baudrate bit (SMOD) in PCON disabled
       TL1 = 0xfd;
       TH1 = 0xfd;           //outoherlaai waarde van baudrate-timer 9600
```

```
	TMOD = 0x20;		//stel timer1 op vir UART
	TCON = 0x49;		//aktiveer timer1
	SCON = 0x50;		//serial mode1: 8bit UART, reception enabled

	//init van EXT INT ; edge triggered
	TCON |= ( (1<<2) | (1<<0) );

	//init Interrupt Enable Register ( UART;exitint1 en 0 ; global)
	IE |= (1<<4) | (1<<2) | (1<<0) | (1<<7);

	while(1) {
		toetsrotstap();			// Pas rotasiemotor aan na nuwe fase
		toetstransstap();		// Pas translasiemotor aan na nuwe fase
		if (UARTserviced == 1) {	// voer Seriele opdrag uit indien 'n opdrag wel
ontvang is
			UARTserviced = 0;
			switch (serialcommand) {
				case 0x00:	//reset system to zero position
					resetsys();
				break;
				case 0x01:	//roteer vinnig voorentoe
					while (UARTserviced != 1) {
						eenrotstapvoort();
						toetsrotstap();
						wagsiklus();
					}
					UARTserviced = 0;
				break;
				case 0x02:	//roteer vinnig truwaarts
					while (UARTserviced != 1) {
						eenrotstaptru();
						toetsrotstap();
						wagsiklus();
					}
					UARTserviced = 0;
				break;
				case 0x03:	//stop rotasie
				break;
				case 0x04:	//transleer voorentoe
					while (UARTserviced != 1) {
						eentransstapvoort();
						toetstransstap();
						wagsiklus();
					}
					UARTserviced = 0;
				break;
				case 0x05:	//transleer truwaarts
					while (UARTserviced != 1) {
						eentransstaptru();
						toetstransstap();
						wagsiklus();
					}
					UARTserviced = 0;
				break;
				case 0x06:	//stop translasie
				break;
				case 0x07:	//zero tellers
					rotteller = 0;
					transteller = 0;
				break;
			}
		}
	}
}
```

Bylae G – Programkode van die beeldvasleggingsprogram

```
//---------------------------------------------------------------------------
// Definisie koplêer van die beeldvasleggingsprogram

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <vfw.h>
#include <Menus.hpp>
#include <ExtCtrls.hpp>

#define PPort 0x378
#define SBuffer 0x3f8
#define LBaudDiv 0x3f8
#define HBaudDiv 0x3f9
#define SLCR 0x3fb

//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published: // IDE-managed Components
    TMainMenu *MainMenu1;
    TMenuItem *File1;
    TMenuItem *Source1;
    TMenuItem *Format1;
    TMenuItem *Preview1;
    TTimer *Timer1;
    TMenuItem *MotorCommands1;
    TMenuItem *ResetSystem1;
    TMenuItem *ZeroCounters1;
    TMenuItem *Rotation1;
    TMenuItem *FFWD1;
    TMenuItem *FBWD1;
    TMenuItem *Translation1;
    TMenuItem *FFWD2;
    TMenuItem *FBWD2;
    TTimer *Timer2;
    TTimer *Timer3;
    TMenuItem *SystemCommands1;
    TMenuItem *StartSystem1;
    TMenuItem *StopSystem1;
    TMenuItem *STOP1;
    TMenuItem *STOP2;
    TMenuItem *STEP1;
    TMenuItem *STEP2;
    TTimer *Timer5;
    TEdit *TimeEdit;
    TLabel *TimeEditLabel;
    TGroupBox *RotateGroupBox;
    TGroupBox *TranlateGroupBox;
    TButton *RotOneStepButton;
    TButton *TransOneStepButton;
    TCheckBox *RotCWCheckBox;
    TCheckBox *TransCWCheckBox;
    TPanel *Panel2;
    TCheckBox *DriverSuccessCheckBox;
    TEdit *DriverNameEdit;
    TEdit *FileNameEdit;
    TButton *GrabFrameButton;
    TGroupBox *SystemGroupBox;
    TCheckBox *CaptureCheckBox;
    TLabel *RotMonitorLabel;
    TLabel *TransMonitorLabel;
    TEdit *RotMonitorEdit;
    TEdit *TransMonitorEdit;
    TLabel *NumFilesLabel;
```

```
    TEdit *NumFilesEdit;
    TButton *SystemOneStepButton;
    TButton *SystemStartButton;
       TEdit *RotperPhotoEdit;
       TLabel *RotperPhotoLabel;
       TEdit *PitchEdit;
       TLabel *PitchLabel;
    TButton *SystemStopButton;
    TEdit *ShutterWaitEdit;
    TLabel *ShutterWaitLabel;
    void __fastcall Source1Click(TObject *Sender);
    void __fastcall Format1Click(TObject *Sender);
    void __fastcall Preview1Click(TObject *Sender);
    void __fastcall GrabFrameButtonClick(TObject *Sender);
    void __fastcall Timer1Timer(TObject *Sender);
    void __fastcall ZeroCounters1Click(TObject *Sender);
    void __fastcall STEP1Click(TObject *Sender);
    void __fastcall STOP1Click(TObject *Sender);
    void __fastcall ResetSystem1Click(TObject *Sender);
    void __fastcall FFWD1Click(TObject *Sender);
    void __fastcall FBWD1Click(TObject *Sender);
    void __fastcall STEP2Click(TObject *Sender);
    void __fastcall STOP2Click(TObject *Sender);
    void __fastcall FFWD2Click(TObject *Sender);
    void __fastcall FBWD2Click(TObject *Sender);
    void __fastcall Timer2Timer(TObject *Sender);
    void __fastcall TransCWCheckBoxClick(TObject *Sender);
    void __fastcall RotCWCheckBoxClick(TObject *Sender);
    void __fastcall Timer5Timer(TObject *Sender);
    void __fastcall NumFilesEditChange(TObject *Sender);
    void __fastcall StartSystem1Click(TObject *Sender);
    void __fastcall StopSystem1Click(TObject *Sender);
    void __fastcall Timer3Timer(TObject *Sender);
    void __fastcall SystemOneStepButtonClick(TObject *Sender);
       void __fastcall RotperPhotoEditChange(TObject *Sender);
       void __fastcall PitchEditChange(TObject *Sender);
    void __fastcall ShutterWaitEditChange(TObject *Sender);
private:     // User declarations
public:             // User declarations
    __fastcall TForm1(TComponent* Owner);
    __fastcall ~TForm1(void);
    void UpdateMonitors(void);
    void SetupSerial(void);
    void CaptureImages(void);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
extern "C" __stdcall char Inp32(int);
extern "C" __stdcall char Out32(int, int);

//---------------------------------------------------------------------------
#endif
```

```
//---------------------------------------------------------------------------
/*
Beeldvasleggings program geskryf vir die Borland C++ Builder Samesteller
Aandryf pulse vir stappermotor aandrywing deur die parallelpoort van die
    PC

    Penuitleg van parallellepoort:
        D2 - na EXTINT0 van ATMEL vir rotasiemot pulssein
        D3 - na EXTINT1 van ATMEL vir translasiemot pulssein
        D4 - na P3.4 van ATMEL vir rotasiemotor rigting
        D5 - na P3.5 van ATMEL vir translasiemot rigting

*/
//---------------------------------------------------------------------------

#include <vcl.h>
#include <vfw.h>
#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
HWND hFormWnd = NULL;
HWND hCapWnd = NULL;
bool DriverConnectStatus = false;
int filecount = 0, RotTeller = 0, TransTeller = 0, timeinsecs = 0;
int numfiles = 1000, rotperphoto, rotpertrans, wagter = 1;
int CapTeller = 0;


//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    char Device[128];
    char Version[128];

    Out32(PPort, (Inp32(PPort) | (1<<4)) ); //Stel RotCWC hoog
    Out32(PPort, (Inp32(PPort) | (1<<5)) ); //Stel TransCWC hoog

    SetupSerial();
    UpdateMonitors();
    NumFilesEdit->Text = numfiles;
    rotperphoto = RotperPhotoEdit->Text.ToInt();
    rotpertrans = PitchEdit->Text.ToInt();

    hFormWnd = Form1->Handle;
    hCapWnd = capCreateCaptureWindow("Capture Window", WS_CHILD|WS_VISIBLE,
                                     5, 5, 640, 480, hFormWnd, 0);

    if( capGetDriverDescription(0, Device, 128, Version, 128) ) {
        DriverNameEdit->Text = Device;
    }

    if ( capDriverConnect(hCapWnd, 0) ) {
        DriverSuccessCheckBox->Checked = true;
        DriverConnectStatus = true;
//        capPreview(hCapWnd, true);
    }
}
//---------------------------------------------------------------------------

__fastcall TForm1::~TForm1(void)
{
    if (DriverConnectStatus) {
```

```
        capDriverDisconnect(hCapWnd);
    }
    DestroyWindow(hCapWnd);
    hCapWnd = NULL;
}
//---------------------------------------------------------------------------

void TForm1::SetupSerial(void)
{
    Out32(SLCR, Inp32(SLCR) | (1<<7) ); //Stel op om baudrate diviser in te lees
    Out32(LBaudDiv, 0x0C);      //Lower byte of divisor for 9600
    Out32(HBaudDiv, 0x00);      //Higher byte of divisor for 9600

    Out32(SLCR, 0x03);      //Line Control Reg vir 8bit,no par,1stop
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Source1Click(TObject *Sender)
{
    if (DriverConnectStatus) {
        capDlgVideoSource(hCapWnd);
    }
    else {
        Application->MessageBox("Driver not Connected", "Error", MB_OK);
    }
}
//---------------------------------------------------------------------------

void TForm1::UpdateMonitors(void)
{
    RotMonitorEdit->Text = RotTeller;
    TransMonitorEdit->Text = TransTeller;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Format1Click(TObject *Sender)
{
    if (DriverConnectStatus) {
        capDlgVideoFormat(hCapWnd);
    }
    else {
        Application->MessageBox("Driver not Connected", "Error", MB_OK);
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Preview1Click(TObject *Sender)
{
    if (DriverConnectStatus) {
        capPreviewRate(hCapWnd, 30);
        capPreview(hCapWnd, true);
    }
    else {
        Application->MessageBox("Driver not Connected", "Error", MB_OK);
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::GrabFrameButtonClick(TObject *Sender)
{
    if (DriverConnectStatus) {
        capGrabFrame(hCapWnd);
        capFileSaveDIB(hCapWnd, "prent.bmp");
//        capPreview(hCapWnd, true);
    }
    else {
        Application->MessageBox("Driver not Connected", "Error", MB_OK);
    }
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    char c;

    if (RotCWCheckBox->Checked == true) {
        ++RotTeller;
        if (RotTeller >= 5000) {
            RotTeller = 0;
        }
    }
    else {
        --RotTeller;
        if (RotTeller < 0) {
            RotTeller = 4999;
        }
    }

    c = Inp32(PPort) | (1<<2);      //maak bit hoog
    Out32(PPort, c);
    Out32(PPort, c & ~(1<<2) );     //maak bit laag

    UpdateMonitors();

}
//---------------------------------------------------------------------------

void __fastcall TForm1::ZeroCounters1Click(TObject *Sender)
{
    // Reset App Counters
    RotTeller = 0;
    TransTeller = 0;
    CapTeller = 0;
    UpdateMonitors();

    //Sent Command to reset ATMEL counters
    Out32(SBuffer, 0x07);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::STEP1Click(TObject *Sender)
{
    Timer1->Enabled = true; //Rotate motor
}
//---------------------------------------------------------------------------

void __fastcall TForm1::STOP1Click(TObject *Sender)
{
    Timer1->Enabled = false;    //Stop Rot. motor
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ResetSystem1Click(TObject *Sender)
{
    Out32(SBuffer, 0x00);   //Command to ATMEL to Reset system

    RotTeller = 0;
    TransTeller = 0;
    UpdateMonitors();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FFWD1Click(TObject *Sender)
{
    Out32(SBuffer, 0x01);   //Command to ATMEL to Rotate FWD
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FBWD1Click(TObject *Sender)
```

```
{
    Out32(SBuffer, 0x02);   //Command to ATMEL to Rotate BWD
}
//---------------------------------------------------------------------------

void __fastcall TForm1::STEP2Click(TObject *Sender)
{
    Timer2->Enabled = true; //Translate motor
    Timer5->Enabled = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::STOP2Click(TObject *Sender)
{
    Timer2->Enabled = false;    //Stop Trans. motor
    Timer5->Enabled = false;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FFWD2Click(TObject *Sender)
{
    Out32(SBuffer, 0x04);   //Command to ATMEL to translate FWD
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FBWD2Click(TObject *Sender)
{
    Out32(SBuffer, 0x05);   //Command to ATMEL to Translate BWD
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
    char c;

    if (TransCWCheckBox->Checked == true) {
        ++TransTeller;
    }
    else {
        --TransTeller;
    }

    c = Inp32(PPort) | (1<<3);      //maak bit hoog
    Out32(PPort, c);
    Out32(PPort, c & ~(1<<3) );     //maak bit laag

    UpdateMonitors();

}
//---------------------------------------------------------------------------

void TForm1::CaptureImages(void)
{
    if (filecount < 10) {
        FileNameEdit->Text = "image000" + IntToStr(filecount) + ".bmp";
    }
    else {
        if (filecount < 100) {
            FileNameEdit->Text = "image00" + IntToStr(filecount) + ".bmp";
        }
        else {
            if (filecount < 1000) {
                FileNameEdit->Text = "image0" + IntToStr(filecount) + ".bmp";
            }
            else {
                FileNameEdit->Text = "image" + IntToStr(filecount) + ".bmp";
            }
        }
    }
```

```
    ++filecount;

    if (DriverConnectStatus) {
        capGrabFrame(hCapWnd);
        capFileSaveDIB(hCapWnd, FileNameEdit->Text.c_str() );
//        capPreview(hCapWnd, true);
    }
    else {
        Application->MessageBox("Driver not Connected", "Error", MB_OK);
    }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::TransCWCheckBoxClick(TObject *Sender)
{
    if (TransCWCheckBox->Checked == true) {
        Out32(PPort, (Inp32(PPort) & ~(1<<5)) );
    }
    else {
        Out32(PPort, (Inp32(PPort) | (1<<5)) );
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::RotCWCheckBoxClick(TObject *Sender)
{
    if (RotCWCheckBox->Checked == true) {
        Out32(PPort, (Inp32(PPort) & ~(1<<4)) );
    }
    else {
        Out32(PPort, (Inp32(PPort) | (1<<4)) );
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Timer5Timer(TObject *Sender)
{
    ++timeinsecs;
    TimeEdit->Text = timeinsecs;
}
//---------------------------------------------------------------------------


void __fastcall TForm1::NumFilesEditChange(TObject *Sender)
{
    if (!NumFilesEdit->Text.IsEmpty()) {
        numfiles = NumFilesEdit->Text.ToInt();
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::StartSystem1Click(TObject *Sender)
{
    Timer3->Enabled = true;
    Timer5->Enabled = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::StopSystem1Click(TObject *Sender)
{
    Timer3->Enabled = false;
    Timer5->Enabled = false;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Timer3Timer(TObject *Sender)
{

    char c;
    Timer3->Enabled = false;
```

```
if (wagter == 1) {  // Wagter if begin
  // Stapmotor kode begin
    if (RotCWCheckBox->Checked == true) {
       ++CapTeller;
        ++RotTeller;

        c = Inp32(PPort) | (1<<2);      //maak bit hoog
        Out32(PPort, c);
        Out32(PPort, c & ~(1<<2) );     //maak bit laag

        if (RotTeller >= 5000) {
            RotTeller = 0;
        }

        if ( (RotTeller % rotpertrans) == 0 ) {
//        if ( ((RotTeller % 33) == 0) || ((RotTeller % 1666) == 0) ) {
            if (TransCWCheckBox->Checked == true) {
                ++TransTeller;
                c = Inp32(PPort) | (1<<3);      //maak bit hoog
                Out32(PPort, c);
                Out32(PPort, c & ~(1<<3) );     //maak bit laag

                if (TransTeller >= 58000) {
                    Timer3->Enabled = false;
                }
            }
            else {
                --TransTeller;
                c = Inp32(PPort) | (1<<3);      //maak bit hoog
                Out32(PPort, c);
                Out32(PPort, c & ~(1<<3) );     //maak bit laag
            }
        }
    }
    else {
       --CapTeller;
        --RotTeller;

        c = Inp32(PPort) | (1<<2);      //maak bit hoog
        Out32(PPort, c);
        Out32(PPort, c & ~(1<<2) );     //maak bit laag

        if (RotTeller < 0) {
            RotTeller = 4999;
        }

        if ( (RotTeller % rotpertrans) == 0 ) {
//        if ( ((RotTeller % 33) == 0) || ((RotTeller % 1666) == 0) ) {
            if (TransCWCheckBox->Checked == true) {
                ++TransTeller;
                c = Inp32(PPort) | (1<<3);      //maak bit hoog
                Out32(PPort, c);
                Out32(PPort, c & ~(1<<3) );     //maak bit laag

                if (TransTeller >= 58000) {
                    Timer3->Enabled = false;
                }
            }
            else {
                --TransTeller;
                c = Inp32(PPort) | (1<<3);      //maak bit hoog
                Out32(PPort, c);
                Out32(PPort, c & ~(1<<3) );     //maak bit laag
            }
        }
    }
    UpdateMonitors();
  // Stapmotor kode eindig
```

```
}      // Wagter if eindig

  // Kamera kode begin
    if ( (CaptureCheckBox->Checked == true) && ((CapTeller % rotperphoto) == 0) ) {
              if (wagter == 1) {
              wagter = 0;
             Timer3->Interval = ShutterWaitEdit->Text.ToInt();
         }
         else {
               CaptureImages();
             wagter = 1;
             Timer3->Interval = 5;
         }
    }
  // Kamera kode eindig

    if (filecount <= numfiles) {
        Timer3->Enabled = true;
    }
    else {
        Timer5->Enabled = false;
    }

}
//---------------------------------------------------------------------------

void __fastcall TForm1::SystemOneStepButtonClick(TObject *Sender)
{
    char c;

    if (RotCWCheckBox->Checked == true) {
       ++CapTeller;
        ++RotTeller;

        c = Inp32(PPort) | (1<<2);       //maak bit hoog
        Out32(PPort, c);
        Out32(PPort, c & ~(1<<2) );      //maak bit laag

        if (RotTeller >= 5000) {
            RotTeller = 0;
        }

        if ( (RotTeller % rotpertrans) == 0 ) {
//        if ( ((RotTeller % 33) == 0) || ((RotTeller % 1666) == 0) ) {
            if (TransCWCheckBox->Checked == true) {
                ++TransTeller;
                c = Inp32(PPort) | (1<<3);       //maak bit hoog
                Out32(PPort, c);
                Out32(PPort, c & ~(1<<3) );      //maak bit laag
            }
            else {
                --TransTeller;
                c = Inp32(PPort) | (1<<3);       //maak bit hoog
                Out32(PPort, c);
                Out32(PPort, c & ~(1<<3) );      //maak bit laag
            }
        }
    }
    else {
       --CapTeller;
        --RotTeller;

        c = Inp32(PPort) | (1<<2);       //maak bit hoog
        Out32(PPort, c);
        Out32(PPort, c & ~(1<<2) );      //maak bit laag

        if (RotTeller < 0) {
            RotTeller = 4999;
        }
```

```
        if ( (RotTeller % rotpertrans) == 0 ) {
//        if ( ((RotTeller % 33) == 0) || ((RotTeller % 1666) == 0) ) {
            if (TransCWCheckBox->Checked == true) {
                ++TransTeller;
                c = Inp32(PPort) | (1<<3);      //maak bit hoog
                Out32(PPort, c);
                Out32(PPort, c & ~(1<<3) );     //maak bit laag
            }
            else {
                --TransTeller;
                c = Inp32(PPort) | (1<<3);      //maak bit hoog
                Out32(PPort, c);
                Out32(PPort, c & ~(1<<3) );     //maak bit laag
            }
        }
    }

    UpdateMonitors();

    if ( (CaptureCheckBox->Checked == true) && ((CapTeller % rotperphoto) == 0) ) {
        CaptureImages();
    }

}
//---------------------------------------------------------------------------
void __fastcall TForm1::RotperPhotoEditChange(TObject *Sender)
{
      if ( !RotperPhotoEdit->Text.IsEmpty() ) {
            rotperphoto = RotperPhotoEdit->Text.ToInt();
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::PitchEditChange(TObject *Sender)
{
      if ( !PitchEdit->Text.IsEmpty() ) {
            rotpertrans = PitchEdit->Text.ToInt();
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ShutterWaitEditChange(TObject *Sender)
{
    if (!ShutterWaitEdit->Text.IsEmpty()) {
    }
}
```

Bylae H – Programkode van die lasersensor-terugvoerstelsel opneemmetodes

```
// Definisie koplêer van die lasersensor-terugvoerstelsel beheer- en opneemprogram
      // Serial (COM1) Constants
#define SBuffer 0x3f8
#define LBaudDiv 0x3f8
#define HBaudDiv 0x3f9
#define SLCR 0x3fb

      // Serial Commands to ATMEL for Steppers
#define CMD_RESET 0x00
#define CMD_ROTFWD 0x01
#define CMD_ROTBWD 0x02
#define CMD_TRANSFWD 0x04
#define CMD_TRANSBWD 0x05
#define CMD_ZERO 0x07


void print_mainmenu(void);
char print_opstellingmenu(void);
char print_stelselmenu(void);
char print_rotasiemenu(void);
char print_translasiemenu(void);
void handle_1(void);
void handle_1_1(void);
void handle_1_2(void);
void handle_1_3(void);
void handle_1_4(void);
void handle_1_5(void);
void handle_1_6(void);
void handle_2(void);
void handle_2_1(void);
void handle_2_2(void);
void handle_2_3(void);
void handle_2_4(void);
void handle_3(void);
void handle_3_1(void);
void handle_3_2(void);
void handle_3_3(void);
void handle_4(void);
void handle_4_1(void);
void handle_4_2(void);
void handle_4_3(void);
void eendwars(int);
void eenopname(int);
void anti_rotlash(void);
void anti_translash(void);
void anti_syslash(void);
void onerotstep(void);
void onetransstep(void);
void set_rotrigting(char);
void set_transrigting(char);
void Toggle_rotrigting(void);
void Toggle_transrigting(void);
void eensysstap(void);
void DoenDAInt(void);
void DAout(int,int);
void TriggerAD(void);
int GetSample(void);
void CalcOutIntegral(int);
void CalcOutIntegral2(int);
void Aquire(void);
void Aquire2(void);
void Aquire3(void);
void init_ASA(void);
void init_serial(void);
void init(void);
void wagsiklus(unsigned int);
```

```
// Beheer- en opneemprogram vir die lasersensor-terugvoerstelselmetode
// Geskryf vir Borland C++ 3.1 Samesteller
// Bevat voorwaartse opname, truwaartse opname en dwarsskandering
/*

      Penuitleg van parallellepoort:
            D2    - na EXTINT0 van ATMEL vir rotasiemot pulssein
            D3    - na EXTINT1 van ATMEL vir translasiemot pulssein
            D4    - na P3.4 van ATMEL vir rotasiemotor rigting
            D5    - na P3.5 van ATMEL vir translasiemot rigting

      Penuitleg vir ASA3.0
            Pen 1  - Vin1 - Intree vir monstering van klanksein nadat
                  gefilter met Krohn-Hite
            Pen 2  - Agnd - Grond
            Pen 6  - Vout1      - Oorspronklike integraal uittree VYFDES_VAN_GROEF
            Pen 7  - Vout0      - Ekstra integraal uittree vir DWARS_OPNAME
            Pen 8  - Vin0 - Integraal se intree (sein afkomstig net na
                  naloop laagdeurlaatfilter

*/
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <dos.h>
#include <stdlib.h>
#include <iostream.h>
#include <fstream.h>
#include "hoof.h"

//Opsies vir #defines
//#define TWINTIG_OPNAMES
#define VYFDES_VAN_GROEF
//#define VOOR_EN_TRU_OPNAME
//#define DWARS_OPNAME
//#define SELFDE_DWARS
//#define NA_INTEGRAAL
//#define NA_INTEGRAAL2

// Sleutel sleutels
#define     ESC         0x1b
#define     CKEY        0x63
#define     HKEY        0x68
#define     JKEY        0x6A
#define     QKEY        0x71
#define     RKEY        0x72
#define     SKEY        0x73
#define     NULKEY      0x30
#define     ONEKEY      0x31
#define     TWOKEY      0x32
#define     THREEKEY    0x33
#define     FOURKEY           0x34
#define     FIVEKEY           0x35
#define     SIXKEY      0x36

// Parallelpoort
#define pport       0x378

// ASA Kaart konstantes
#define ADbase            0x360
// afset van basisadres
#define     ADlow       (ADbase+0x00)
#define     ADhigh      (ADbase+0x02)
#define     DA0low      (ADbase+0x04)
#define     DA0high           (ADbase+0x05)
#define     DA1low      (ADbase+0x06)
#define     DA1high     (ADbase+0x07)
#define     ADcount0    (ADbase+0x08)
```

```
#define      ADcount1      (ADbase+0x09)
#define      ADcount2      (ADbase+0x0A)
#define      ADcntctrl     (ADbase+0x0B)
#define      ADporta             (ADbase+0x0C)
#define      ADportb             (ADbase+0x0D)
#define      ADportc             (ADbase+0x0E)
#define      ADportctrl    (ADbase+0x0F)

// Macro's
#define hi(x)        ( ((x)>>8) & 0x00FF )      // high byte
#define lo(x)        ((x) & 0x00FF)             // low byte

//global var's
unsigned int waitcount = 26900;
char rotrigting, transrigting,bufferchanged=1;
int rotpos=0,rotpertrans=33,rotpersample=1,transpermod=1;
int ISample,Integral=0,DA0wde=0,DA1wde=0,Ki=8;
long transpos=0,totpos=0,bufferlen=10000,aantal_rots=0;
int lockcount=0;
char lock = 0;

// Hoof program menu
void print_mainmenu(void) {
      cout << endl << " Kies 'n nommer :" << endl;
      cout << "   1. Opstelling" << endl;
      cout << "   2. Stelsel" << endl;
      cout << "   3. Rotasiestelsel" << endl;
      cout << "   4. Translasiestelsel" << endl;
}

// Menu 1 - Opstelling menu
// vertoon opsien en wag vir keuse
char print_opstellingmenu(void) {
      char c;

      clrscr();
      cout << endl << " -Opstelling-" << endl;
      cout << "   1. Verstel wagsikluswaarde" << endl;
      cout << "   2. Naam van datale^r" << endl;
      cout << "   3. Pitch (Rotasiestappe per translasiestap)" << endl;
      cout << "   4. Rotasie stappe per monster" << endl;
      cout << "   5. Translasiestappe per rotasiemodulus;" << endl;
      cout << "   6. Aantal monsterpunte vir opname (grootte van datary)" << endl;

      c = getch(); // wag vir keuse
      return c;
}

// Menu 2 - Stelsel menu
// Vertoon opsies en wag vir keuse
char print_stelselmenu(void) {
      char c;

      clrscr();
      cout << endl << " -Stelsel-" << endl;
      cout << "   1. Herstel stelsel" << endl;
      cout << "   2. Zero tellers" << endl;
      cout << "   3. Begin opname" << endl;
      cout << "   4. Geslote lus" << endl;
      cout << "   5. Doen \"Aquire\"." << endl;

      c = getch();
      return c;
}

// Menu 3 - Rotasie menu
// vertoon opsie en wag vir keuse
char print_rotasiemenu(void) {
```

```
	char c;

	clrscr();
	cout << endl << " -Rotasiestelsel-" << endl;
	cout << "  Kies 'n nommer :" << endl;
	cout << "   1. Een Rotasiestap" << endl;
	cout << "   2. Roteer" << endl;
	cout << "   3. Rotasierigting" << endl;

	c = getch();
	return c;
}

// Menu 4 - Translasie menu
// Vertoon opsies en wag vir keuse
char print_translasiemenu(void) {
	char c;

	clrscr();
	cout << endl << " -Translasiestelsel-" << endl;
	cout << "  Kies 'n nommer :" << endl;
	cout << "   1. Een Translasiestap" << endl;
	cout << "   2. Transleer" << endl;
	cout << "   3. Translasierigting" << endl;

	c = getch();
	return c;
}

// Submenu 1.1 - Verstel wagsikluswaarde
void handle_1_1(void) {
	cout << endl << "   Huidige wagsikluswaarde: " << waitcount << endl;
	cout << endl << "   Sleutel nuwe waarde in: ";
	cin >> waitcount;
}

// Submenu 1.2 - Verstel dataleernaam
void handle_1_2(void) {
//	cout << endl << "   Huidige le^r: " << dataleernaam;
//	cout << endl << "   Naam van datale^r? ";
//	cin >> dataleernaam;
}

// Submenu 1.3 - Verstel pitch
void handle_1_3(void) {
	cout << endl << "   Huidige \"pitch\"-waarde: " << rotpertrans << endl;
//	cout << "    0 =  << endl;
	cout << "    33 = is ongeveer pitch van die groef" << endl;
	cout << "    5000 = voltooi een rotasie voor 'n translasie" << endl;
	cout << "   Sleutel nuwe waarde in: ";
	cin >> rotpertrans;
}

// Submenu 1.4 - Verstel rotasiestappe per monster
void handle_1_4(void) {
	cout << endl << "   Huidige \"rotasiestappe per monster\"-waarde: ";
	cout << rotpersample << endl;
	cout << "   Sleutel nuwe waarde in: ";
	cin >> rotpersample;
}

// Submenu 1.5 - Verstel translasiestappe per rotasiemodulus
void handle_1_5(void) {
	cout << endl << "   Huidige \"translasiestappe per rotasiemodulus\"-waarde: ";
	cout << transpermod << endl;
	cout << "   Sleutel nuwe waarde in: ";
	cin >> transpermod;
}
```

```
// Aantal datapunte om te monster (grootte van datary)
void handle_1_6(void) {
	cout << endl << "   Huidige grootte van datary: ";
	cout << bufferlen << endl;
	cout << "   Sleutel nuwe waarde in: ";
	cin >> bufferlen;
	bufferchanged = 1;
}

// Hanteer menu 1 opsies
void handle_1() {
	switch (print_opstellingmenu()) {
		case ONEKEY:
			handle_1_1();
		break;
		case TWOKEY:
			handle_1_2();
		break;
		case THREEKEY:
			handle_1_3();
		break;
		case FOURKEY:
			handle_1_4();
		break;
		case FIVEKEY:
			handle_1_5();
		break;
		case SIXKEY:
			handle_1_6();
		break;
	}
}

// Submenu 2.1 - Herstel stelsel
void handle_2_1(void) {
//	outp(SBuffer, CMD_RESET);   //Command to ATMEL to Reset system

//	Toggle die rotasie en translasie rigtings en neem stappe totdat
//		posisietellers nul is, en voer dan anti-backlash uit
	char done = 0;
	unsigned int tempwait = waitcount;

	waitcount = 3500;

	Toggle_rotrigting();
	Toggle_transrigting();

	while (!done) {
		if (rotpos != 0) {
			onerotstep();
			wagsiklus(waitcount);
		}

		if (transpos != 0) {
			onetransstep();
			wagsiklus(waitcount);
		}

		if ((rotpos == 0) && (transpos == 0)) {
			done = 1;
		}
	}
	waitcount = tempwait;

	Toggle_rotrigting();
	Toggle_transrigting();
```

```
        anti_syslash();             // 200 stappe terug
        anti_syslash();             // 200 stappe voorentoe

//      handle_2_2();// Zero al die tellers

//      rotpos = 0;
//      transpos = 0;
}

// Submenu 2.2 - Zero Tellers
void handle_2_2(void) {
        // Reset App Counters
        rotpos = 0;
        transpos = 0;
        totpos = 0;

        //Sent Command to reset ATMEL counters
        outp(SBuffer, CMD_ZERO);
}

// Submenu 2.3 - Begin opname
// Bevat opname prosedure
void handle_2_3(void) {
#ifdef NA_INTEGRAAL2
        int done = 0,temp,teller=0;
//      waitcount = 3500;

        handle_2_1();// reset stelsel en anti_lash
        handle_2_2();

        while (!done) {             // rotasie while
                gotoxy(60,16);
                cout << "                ";
                gotoxy(60,16);
                cout << "Rotpos: " << rotpos;

                if (teller >= 3) {
                        done = 1;
                }

                eendwars(teller++);

//              onerotstep();

                Toggle_transrigting();
                while (transpos != 0) {
                        onetransstep();
                        wagsiklus(4000);
                }
                Toggle_transrigting();
                anti_lash();
                anti_lash();

// integraalkorreksie
                for (int k=1;k<400;k++) {
                        wagsiklus(3500);
                        TriggerAD();
                        temp = GetSample();
                        CalcOutIntegral(ISample);
                }

                if (kbhit())
                        if (getch() == ESC)
                                done = 1;


        }       // rotasie while
```

```
#endif

#ifdef NA_INTEGRAAL
	int done = 0,temp,teller=0;
//	waitcount = 3500;

	handle_2_1();// reset stelsel en anti_lash
	handle_2_2();

	while (!done) {			// rotasie while
		gotoxy(60,16);
		cout << "                ";
		gotoxy(60,16);
		cout << "Rotpos: " << rotpos;

		if (teller >= 3) {
			done = 1;
		}

		eendwars(teller++);

//		onerotstep();

		Toggle_transrigting();
		while (transpos != 0) {
			onetransstep();
			wagsiklus(3500);
		}
		Toggle_transrigting();
		anti_lash();
		anti_lash();

// integraalkorreksie
		for (int k=1;k<400;k++) {
			wagsiklus(3500);
			TriggerAD();
			temp = GetSample();
			CalcOutIntegral(ISample);
		}

		if (kbhit())
			if (getch() == ESC)
				done = 1;


	}	// rotasie while

#endif

#ifdef SELFDE_DWARS
	int done = 0,temp,teller=0;
	waitcount = 6000;

	handle_2_1();// reset stelsel en anti_lash
	handle_2_2();

	while (!done) {			// rotasie while
		gotoxy(60,16);
		cout << "                ";
		gotoxy(60,16);
		cout << "Rotpos: " << rotpos;

		if (teller >= 4999) {
			done = 1;
		}

		eendwars(teller++);
```

```
//              onerotstep();
                delay(200);

                Toggle_transrigting();
                while (transpos != 0) {
                        onetransstep();
                        wagsiklus(5000);
                }
                Toggle_transrigting();
                anti_translash();
                delay(200);
                anti_translash();

// integraalkorreksie
                for (int k=1;k<400;k++) {
                        wagsiklus(3000);
                        TriggerAD();
                        temp = GetSample();
                        CalcOutIntegral(ISample);
                }

                if (kbhit())
                        if (getch() == ESC)
                                done = 1;


        }       // rotasie while

#endif

#ifdef DWARS_OPNAME
        int done = 0,temp;
        waitcount = 12000;

        handle_2_1();// reset stelsel en anti_lash
        handle_2_2();

        while (!done) {             // rotasie while
                gotoxy(60,16);
                cout << "                ";
                gotoxy(60,16);
                cout << "Rotpos: " << rotpos;

                if (rotpos >= 4999) {
                        done = 1;
                }

                eendwars(rotpos);

//              for (int ii=0;ii<100;ii++) {
                        onerotstep();
//                      wagsiklus(waitcount);
//              }

                delay(200);

                Toggle_transrigting();
                while (transpos != 0) {
                        onetransstep();
                        wagsiklus(5000);
                }
                Toggle_transrigting();
                anti_translash();
                delay(200);
                anti_translash();

// integraalkorreksie
                for (int k=1;k<400;k++) {
```

```
                wagsiklus(3000);
                TriggerAD();
                temp = GetSample();
                CalcOutIntegral(ISample);
        }

        if (kbhit())
                if (getch() == ESC)
                        done = 1;


    }       // rotasie while

#endif

#ifdef TWINTIG_OPNAMES
    for (int ss=0;ss<23;ss++) {
        gotoxy(60,12);
        cout << ss;

        handle_2_1();// reset stelsel en anti_lash
        totpos = 0;

        gcvt((float) ss,5,numstr);
        strcpy(dataleernaam,"dat");
        strcat(dataleernaam,numstr);
        strcat(dataleernaam,".raw");

        eenopname();
    }//for
#endif

#ifdef VYFDES_VAN_GROEF
    handle_2_1();// reset stelsel en anti_lash
    handle_2_2();

    for (int xx=0;xx<5;xx++) {
        gotoxy(60,12);
        cout << xx;

        eenopname(xx);

        handle_2_1();//reset stelsel en anti_lash

        for (int aa=0;aa<30;aa++) {       // transleer 'n 5de groefbreedte
            onetransstep();
            wagsiklus(waitcount);
        }//for
        handle_2_2();
    }//for
#endif

#ifdef VOOR_EN_TRU_OPNAME
    handle_2_1();// reset stelsel en anti_lash
    totpos = 0;

    for (int xx=0;xx<5;xx++) {
        gotoxy(60,12);
        cout << xx;

        gcvt((float) xx,5,numstr);
        strcpy(dataleernaam,"dat");
        strcat(dataleernaam,numstr);
        strcat(dataleernaam,"f.raw");

        eenopname(xx);

        Toggle_rotrigting();       //verander rigting truwaarts en anti_lash
```

```
            Toggle_transrigting();
            anti_lash();
            anti_lash();

            handle_2_2();// Zero tellers

            gcvt((float) xx,5,numstr);
            strcpy(dataleernaam,"dat");
            strcat(dataleernaam,numstr);
            strcat(dataleernaam,"b.raw");

            eenopname();

            Toggle_rotrigting();       //verander rigting voorwaarts en anti_lash
            Toggle_transrigting();
            anti_lash();
            anti_lash();

            handle_2_2();//Zero tellers

            for (int aa=0;aa<30;aa++) {       // transleer 'n 5de groefbreedte
                  onetransstep();
                  wagsiklus(waitcount);
            }//for
            transpos = 0;
      }//for

#endif

}//handle_2_3

// Submenu 2.4 - Sluit die lus
void handle_2_4(void) {
      int done = 0,temp;

      while (!done) {
            if (kbhit())
                  if (getch() == ESC)
                        done = 1;

            TriggerAD();
            wagsiklus(waitcount);
            temp = GetSample();
            CalcOutIntegral(ISample);
            if (lock == 0) {
                  Aquire();
            }
      }
}

// Submenu 2.5 - Doen Aquire
void handle_2_5(void) {
      Aquire();
}

// Hanteer menu 2 opsies
void handle_2(void) {
      switch (print_stelselmenu()) {
            case ONEKEY:
                  handle_2_1();
            break;
            case TWOKEY:
                  handle_2_2();
            break;
            case THREEKEY:
                  handle_2_3();
            break;
            case FOURKEY:
```

```
                handle_2_4();
            break;
            case FIVEKEY:
                handle_2_5();
            break;
        }
}

// Submenu 3.1 - Een rotasiestap vir elke keer wat '1'-sleutel gedruk word
void handle_3_1(void) {
        int done = 0;

        onerotstep();

        while (!done) {
            if (kbhit()) {
                switch (getch()) {
                    case ONEKEY:
                        onerotstep();
                    break;
                    case ESC:
                        done = 1;
                    break;
                }       //switch
            }       //if
        }       //while
}

// Submenu 3.2 - roteer teen wagsiklusspoed totdat ESC gedruk word
void handle_3_2(void) {
        int done = 0;

        while (!done) {
            onerotstep();
            if (kbhit())
                if (getch() == ESC)
                    done = 1;

            wagsiklus(waitcount);
        }       //while
}

// Submenu 3.3 - Verander rotasierigting
void handle_3_3(void) {
        cout << endl << "   Huidige rotasierigting: ";
        if ( inp(pport) & (1<<4) ) {
            rotrigting = 1;
            cout << " voorwaarts" << endl;
        }
        else {
            rotrigting = 0;
            cout << " truwaarts" << endl;
        }

        cout << "   Verander(J/N)?";
        if (getch() == JKEY) {
            Toggle_rotrigting();
        }

}

// Hanteer menu 3 opsies
void handle_3(void) {
        switch (print_rotasiemenu()) {
            case ONEKEY:
                handle_3_1();
            break;
            case TWOKEY:
```

```
                handle_3_2();
            break;
            case THREEKEY:
                handle_3_3();
            break;
            default:
                clrscr();
                print_mainmenu();
            break;
        }       //switch

}

// Submenu 4.1 - Een translasiestap vir elke keer wat '1'-sleutel gedruk word
void handle_4_1(void) {
        int done = 0;

        onetransstep();

        while (!done) {
            if (kbhit()) {
                switch (getch()) {
                    case ONEKEY:
                        onetransstep();
                    break;
                    case ESC:
                        done = 1;
                    break;
                }       //switch
            }       //if
        }       //while
}

// Submenu 4.2 - transleer teen wagsiklusspoed
void handle_4_2(void) {
        int done = 0;

        while (!done) {
//      for (int a=0;a<10000;a++) {
            onetransstep();
            if (kbhit())
                if (getch() == ESC)
                    done = 1;

            wagsiklus(waitcount);
        }       //while
}

// Submenu 4.3 - Verander translasie rigting
void handle_4_3(void) {
        cout << endl << "   Huidige translasierigting: ";
        if ( inp(pport) & (1<<5) ) {
            transrigting = 1;
            cout << " voorwaarts" << endl;
        }
        else {
            transrigting = 0;
            cout << " truwaarts" << endl;
        }

        cout << "   Verander(J/N)?";
        if (getch() == JKEY) {
            Toggle_transrigting();
        }

}

// Hanteer menu 4 opsies
```

```
void handle_4(void) {
	switch (print_translasiemenu()) {
		case ONEKEY:
			handle_4_1();
		break;
		case TWOKEY:
			handle_4_2();
		break;
		case THREEKEY:
			handle_4_3();
		break;
	}
}

void eendwars(int teller) {
	FILE *out;
	FILE *intout;
	int done2 = 0,temp;
	char numstr[10];
	char dataleernaam[64],intgrnaam[64];

	gcvt((float) teller,5,numstr);
	strcpy(dataleernaam,"ddat");
	strcat(dataleernaam,numstr);
	strcat(dataleernaam,".raw");
	out = fopen(dataleernaam,"wb");

	strcpy(intgrnaam,"int");
	strcat(intgrnaam,numstr);
	strcat(intgrnaam,".raw");
	intout = fopen(intgrnaam,"wb");

	done2=0;
	while (!done2) {	//translasie while
		gotoxy(60,15);
		cout << "                    ";
		gotoxy(60,15);
		cout << "Transpos: " << transpos;

		if (kbhit())
			if (getch() == ESC)
				done2 = 1;

		if (lock == 0) {
			Aquire3();
		}

		TriggerAD();
		temp = GetSample();

		fputc(lo(temp),out);
		fputc(hi(temp),out);
		fputc(lo(Integral),intout);
		fputc(hi(Integral),intout);

		CalcOutIntegral(ISample);

		wagsiklus(waitcount);

		onetransstep();

		if (abs(transpos) >= bufferlen)
			done2 = 1;
	}	// translasie while
	fclose(out);
	fclose(intout);

}
```

```
void eenopname(int leertel) {
	FILE *out;
	int done = 0,temp;
	char numstr[10];
	char dataleernaam[128];

	gcvt((float) leertel,5,numstr);
	strcpy(dataleernaam,"dat");
	strcat(dataleernaam,numstr);
	strcat(dataleernaam,".raw");

	out = fopen(dataleernaam,"wb");

	while (!done) {
		gotoxy(60,16);
		cout << "            ";
		gotoxy(60,16);
		cout << "Totpos: " << totpos;

		if (totpos >= bufferlen) {
			done = 1;
		}

		if (kbhit())
			if (getch() == ESC)
				done = 1;

		if ( totpos % rotpersample == 0 ) {
			TriggerAD();
			temp = GetSample();
//			datauit << temp << endl;
//			fprintf(out,"%u",temp);

			fputc(lo(temp),out);
			fputc(hi(temp),out);

			CalcOutIntegral(ISample);

			if (lock == 0) {
				Aquire();
			}
		}

		eensysstap();
		wagsiklus(waitcount);
	}	//while

//	datauit.close();
	fclose(out);
}

////////////////////////////////////////////////////////
// Anti-backlash roetine
////////////////////////////////////////////////////////
void anti_rotlash(void) {
	unsigned int tempwait = waitcount;
	waitcount = 5000;

	Toggle_rotrigting();

	for (int x=0;x<300;x++) {
		onerotstep();
		wagsiklus(waitcount);
	}
	waitcount = tempwait;
}
```

```
void anti_translash(void) {
	unsigned int tempwait = waitcount;
	waitcount = 5000;

	Toggle_transrigting();

	for (int x=0;x<300;x++) {
		onetransstep();
		wagsiklus(waitcount);
	}
	waitcount = tempwait;
}

void anti_syslash(void) {
	unsigned int tempwait = waitcount;
	waitcount = 5000;

	Toggle_rotrigting();
	Toggle_transrigting();

	for (int x=0;x<300;x++) {
		onerotstep();
		onetransstep();
		wagsiklus(waitcount);
	}
	waitcount = tempwait;
}


////////////////////////////////////////////////////////
// Puls D2 van parallelpoort om een rotasiestap te gee
////////////////////////////////////////////////////////
void onerotstep(void) {
	++totpos;
	if ( inp(pport) & (1<<4) ) {
		--rotpos;
	}
	else {
		++rotpos;
	}

	if (rotpos >= 5000) {
		rotpos = 0;
		++aantal_rots;
	}

	if (rotpos < 0) {
		rotpos = 4999;
		--aantal_rots;
	}

	outp(pport, inp(pport) & ~(1<<2) );	// maak lyn laag
	outp(pport, inp(pport) | (1<<2) );	// maak lyn hoog
}

////////////////////////////////////////////////////////
//	Puls D3 van parallelpoort om een translasiestap te gee
////////////////////////////////////////////////////////
void onetransstep(void) {
	if ( inp(pport) & (1<<5) ) {
		--transpos;
	}
	else {
		++transpos;
	}

	outp(pport, inp(pport) & ~(1<<3) );	// maak lyn hoog
	outp(pport, inp(pport) | (1<<3) );	// maak lyn laag
```

```
}

//////////////////////////////////////////////////////
//     Rotasierigting soos bepaal deur D4 van parallelpoort
//////////////////////////////////////////////////////
void set_rotrigting(char rigting) {
      if (rigting == 1) {
            outp(pport, inp(pport) | (1<<4) );
      }

      if (rigting == 0) {
            outp(pport, inp(pport) & ~(1<<4) );
      }
}

//////////////////////////////////////////////////////
//     Translasierigting soos bepaal deur D5 van parallelpoort
//////////////////////////////////////////////////////
void set_transrigting(char rigting) {
      if (rigting == 1) {
            outp(pport, inp(pport) | (1<<5) );
      }

      if (rigting == 0) {
            outp(pport, inp(pport) & ~(1<<5) );
      }
}

//////////////////////////////////////////////////////////
// Een stap vir die opgestelde stelsel
//////////////////////////////////////////////////////////
void eensysstap(void) {
      // Stapmotor kode begin
      onerotstep();

      if ( (rotpos % rotpertrans) == 0 ) {
            for (int x=0;x<transpermod;x++) {       // indien meer translasiestappe
                  onetransstep();                                 //   per   keer   gegee
moet word
                  if (transpermod != 1) {
                        wagsiklus(waitcount);
                  }
            }

            if ( (rotpos == 0) && ( (aantal_rots % 5) == 0) ) {
                  cout << "herstel";
                  wagsiklus(waitcount);
                  onetransstep();
                  wagsiklus(waitcount);
                  onetransstep();
            }
      }

      gotoxy(60,2);
      cout << "              ";
      gotoxy(60,2);
      cout << rotpos << " " << transpos;

}
//////////////////////////////////////////////////////////////

void Toggle_rotrigting(void) {
      if (rotrigting == 1) {
            set_rotrigting(0);
//          cout << "truwaarts" << endl;
            rotrigting = 0;
      }
      else if (rotrigting == 0) {
```

```
                set_rotrigting(1);
//              cout << "voorwaarts" << endl;
                rotrigting = 1;
        }
        else {
                rotrigting = 1;
        }
}

void Toggle_transrigting(void) {
        if (transrigting == 1) {
                set_transrigting(0);
//              cout << "truwaarts" << endl;
                transrigting = 0;
        }
        else if (transrigting == 0) {
                set_transrigting(1);
//              cout << "voorwaarts" << endl;
                transrigting = 1;
        }
        else {
                transrigting = 1;
        }
}

void DoenDAInt(void) {
        DA0wde = Integral/2;
        if (Integral % 2)
                DA1wde = Integral/2+1;
        else
                DA1wde = Integral/2;

        gotoxy(60,5);
        cout << "                ";
        gotoxy(60,5);
        cout << "DA0wde : " << DA0wde;
        gotoxy(60,6);
        cout << "                ";
        gotoxy(60,6);
        cout << "DA1wde : " << DA1wde;
}

////////////////////////////////////////////////////
//      Plaas 12bit integerwaarde uit op D/A.
//      12bit int moet tussen 2047 en -2048 wees.
//      Die "xxxx 0000 0000 0000" (-2048(12bit)) stel die
//    minimum waarde (-10V) voor en "xxxx 1111 1111 1111"
//    (2047(12bit) stel die maksimum waarde (+10V) voor
//       dus moet 'n "sagteware afset" van 2048 (0x0800) by
//       die uittreewaarde bygetel word om dit integerwaarde
//    in die regte formaat te kry.
////////////////////////////////////////////////////
void DAout(int DAnum, int outval) {
        char dummy;
        static int outcount=0;

        if ((outval > 2047) || (outval < -2048)) {
                gotoxy(1,12);
                cout << "D/A waarde " << outcount++ << " buite grense" << endl;
//              getch();
                return;
        }

        outval = outval + 0x0800; //     Tel die sagtewareafset by
        if (DAnum == 0) {
                outp(DA0high, hi(outval) );
                outp(DA0low, lo(outval) );
                dummy = inp(DA0low);
```

```
        }

        if (DAnum == 1) {
                outp(DA1high, hi(outval) );
                outp(DA1low, lo(outval) );
                dummy = inp(DA1low);
        }
}

// Start Conversion
void TriggerAD(void) {
        outp(ADbase,0x00);
}

////////////////////////////////////////////////
//      Lees monsterpunt in deur A/D
////////////////////////////////////////////////
int GetSample(void) {
        char dummy;
        int SSample;

        do {   // wag tot einde van "conversion"
                outp(ADhigh,0x00);
        } while ( inp(ADhigh) & (1<<7) );

        //leer in vanaf 4 A/D's
        dummy = inp(ADlow);
        ISample = (inp(ADhigh)<<8) | dummy;      // Vanaf Vin0
        dummy = inp(ADlow);
        SSample = (inp(ADhigh)<<8) | dummy;      // Vanaf Vin1
        dummy = inp(ADlow);
        dummy = inp(ADhigh);
        dummy = inp(ADlow);
        dummy = inp(ADhigh);

        // ISample en FSample is 12-bit 2's kompliment getalle EN bit 15
        // van die ingeleesde waarde is altyd '1' (???).
        // Die getalle moet nou eers omgeskakel word na 16-bit
        // 2's kompliment

        ISample &= ~(1<<15);
        SSample &= ~(1<<15);

        if ((ISample > 2049) && (ISample < 4096)) {
                ISample -= 4096;
        }

        if (ISample == 2048) {
                ISample = -2047;
        }

        if ((SSample > 2049) && (SSample < 4096)) {
                SSample -= 4096;
        }

        if (SSample == 2048) {
                SSample = -2047;
        }

        gotoxy(60,18);
        cout << "                    ";
        gotoxy(60,18);
        cout << "ISample: " << ISample;

        gotoxy(60,19);
        cout << "                    ";
        gotoxy(60,19);
        cout << "SSample: " << SSample;
```

```
	if ((ISample > 2047) || (ISample < -2048)) {
		gotoxy(60,20);
		cout << "Something Fishy #1";
	}

	if ((SSample > 2047) || (ISample < -2048)) {
		gotoxy(60,21);
		cout << "Something Fishy #2";
	}

	return SSample;
}

void CalcOutIntegral(int Monster) {

	Integral = Integral + (Monster/156);//150

	gotoxy(60,17);
	cout << "                    ";
	gotoxy(60,17);
	cout << "Integral: " << Integral;

//	if (Integral > 4095)
//		Integral = 2046;
//		Aquire2();

//	if (Integral < -4095)
//		Integral = -2046;
//		Aquire2();

	if ((ISample > -50) && (ISample < 50)) {
		++lockcount;
		if (lockcount > 200)
			lock = 0;
	}
	else
		lockcount = 0;

	gotoxy(60,14);
	cout << "                ";
	gotoxy(60,14);
	cout << "Lockcount: " << lockcount;

#ifdef VYFDES_VAN_GROEF
	DAout(1,Integral);
#endif

#ifdef DWARS_OPNAME
	DoenDAInt();
//	DAout(1,Integral/Ki);
	DAout(1,DA1wde/Ki);
	DAout(0,DA0wde/Ki);
#endif

}

// Kry Lock van laser
void Aquire(void) {
	static int dropout=0;
	int k=1999,temp;
	char done = 0;

	++dropout;
	gotoxy(60,3);
	cout << "ReAquire no.: " << dropout;

	DAout(1,k);
```

```
	delay(500);				// wagsiklus van 500ms

	do {	// Hierdie lus moet gebreek word as ISample 'n grootwaarde het
		k-=4;
		DAout(1,k);
		TriggerAD();
		wagsiklus(waitcount);
		temp = GetSample();
		// ISample is ook gemonster deur GetSample
		gotoxy(60,4);
		cout << "              ";
		gotoxy(60,4);
		cout << "A/D: " << ISample;
		gotoxy(60,5);
		cout << "          ";
		gotoxy(60,5);
		cout << "k: " << k;

		if (kbhit()) {
			if (getch() == ESC) {
				done = 1;
			}
		}

	} while ( (ISample > -300) && (ISample < 200) && (k > -1900) && (!done));

	Integral = 0;

	for (k=1;k<600;k++) {
		wagsiklus(waitcount/4);
		TriggerAD();
		temp = GetSample();
		CalcOutIntegral(ISample);
	}
	lockcount = 0;
	lock = 1;
}

// Kry Lock van laser
// Hier word D/A0 saam met D/A1 gebruik om resolusie te vermeerder
void Aquire2(void) {
	static int dropout=0;
	int k=1999,temp;
	char done = 0;

	++dropout;
	gotoxy(60,3);
	cout << "ReAquire no.: " << dropout;

	if (Integral>2048)
		DAout(1,(Integral-1000)/Ki);
	else if (Integral<-2048)
		DAout(1,(Integral+1000)/Ki);
	else
		DAout(1,500/Ki);

	DAout(0,k);
	delay(200);				// wagsiklus van 200ms

	do {	// Hierdie lus moet gebreek word as ISample 'n grootwaarde het
		k-=2;
		DAout(0,k);
		TriggerAD();
		wagsiklus(waitcount);
		temp = GetSample();
		// ISample is ook gemonster deur GetSample
		gotoxy(60,4);
		cout << "              ";
```

```
            gotoxy(60,4);
            cout << "A/D: " << ISample;
            gotoxy(60,5);
            cout << "          ";
            gotoxy(60,5);
            cout << "k: " << k;

            if (kbhit()) {
                  if (getch() == ESC) {
                        done = 1;
                  }
            }

      } while ( (ISample > -500) && (ISample < 500) && (k > -1900) && (!done));

      int SubInt=k;
      for (k=1;k<500;k++) {
            wagsiklus(waitcount/2);
            TriggerAD();
            temp = GetSample();
//          CalcOutIntegral(ISample);
            SubInt = SubInt + ISample/100;
            DAout(0,SubInt);
      }
      lockcount = 0;
      lock = 1;

      if (kbhit())
            if (getch()==ESC)
                  return;

      Aquire();

}

// Kry Lock van laser
// Hier word D/A0 saam met D/A1 gebruik om resolusie te vermeerder
void Aquire3(void) {
      static int dropout=0;
      int temp;
      char done = 0;

      ++dropout;
      gotoxy(60,3);
      cout << "ReAquire no.: " << dropout;

      Integral = 30000;
      DoenDAInt();
      DAout(0,DA0wde/Ki);
      DAout(1,DA1wde/Ki);
      delay(200);                  // wagsiklus van 200ms

      do {  // Hierdie lus moet gebreek word as ISample 'n grootwaarde het
            Integral-=(2*Ki);
            DoenDAInt();
            DAout(0,DA0wde/Ki);
            DAout(1,DA1wde/Ki);
            TriggerAD();
            wagsiklus(waitcount);
            temp = GetSample();
            // ISample is ook gemonster deur GetSample
            gotoxy(60,4);
            cout << "              ";
            gotoxy(60,4);
            cout << "A/D: " << ISample;
            gotoxy(60,5);
            cout << "          ";
            gotoxy(60,5);
```

```
            cout << "DA0wde: " << DA0wde;
            gotoxy(60,6);
            cout << "          ";
            gotoxy(60,6);
            cout << "DA0wde: " << DA0wde;

            if (kbhit()) {
                  if (getch() == ESC) {
                        done = 1;
                  }
            }

      } while ( (ISample > -500) && (ISample < 500) && (Integral > -30000) && (!done));

      for (int k=1;k<500;k++) {
            wagsiklus(waitcount/2);
            TriggerAD();
            temp = GetSample();
            CalcOutIntegral(ISample);
      }
      lockcount = 0;
      lock = 1;
}

////////////////////////////////////////////////
// skryf '1000 0000 0000' = 0x0800 uit na D/A wat
//           gelyk is aan 0V
////////////////////////////////////////////////
void init_ASA(void) {
      DAout(0,0);
      DAout(1,0);
}

void init_serial(void) {
      outp(SLCR, inp(SLCR) | (1<<7) ); //Stel op om baudrate diviser in te lees
      outp(LBaudDiv, 0x0C);      //Lower byte of divisor for 9600
      outp(HBaudDiv, 0x00);      //Higher byte of divisor for 9600

      outp(SLCR, 0x03);      //Line Control Reg vir 8bit,no par,1stop
}

// Inisialiseering
void init(void) {
      clrscr();     //clear screen
      print_mainmenu();

      rotrigting = 0;
      set_rotrigting(rotrigting);
      transrigting = 0;
      set_transrigting(transrigting);

      init_serial();
      cout << "Serieel inisialiseer" << endl;
      init_ASA();
      cout << "ASA inisialiseer" << endl;
}

// Wagsiklus met ongeveer 0.566us/a-waarde
//  Wagsiklus waarde (var: waitcount) is gekies as
//    (1/52(Hz)) / (0.566) = 33977
void wagsiklus(unsigned int tel) {
      long double X;
      unsigned int a,b=100;

      for (a=0;a<tel;a++) {
            for (unsigned int xx=0;xx<b;xx++) {
                  X = 100;
                  X *= X;
```

```
            }
        }
}

void main(void) {
        int done = 0;
        char c;

        init();

        while (!done) {

                if (kbhit()) {
                        switch (c = getch()) {
                                case ESC:
                                        cout << endl << "Verlaat (J/N)?";
                                        if (getch() == JKEY)
                                                done = 1;
                                        else {
                                                clrscr();
                                                print_mainmenu();
                                        }
                                break;
                                case ONEKEY:
                                        handle_1();
                                        clrscr();
                                        print_mainmenu();
                                break;
                                case TWOKEY:
                                        handle_2();
//                                      clrscr();
//                                      print_mainmenu();
                                break;
                                case THREEKEY:
                                        handle_3();
                                        clrscr();
                                        print_mainmenu();
                                break;
                                case FOURKEY:
                                        handle_4();
                                        clrscr();
                                        print_mainmenu();
                                break;
                                case QKEY:
                                {
                                        FILE *Fintg;
                                        FILE *Fsens;
                                        FILE *Fsein;
                                        int intgrwde=0,donee=0,temp1;

                                        Fintg = fopen("intgwb.raw","wb");
                                        Fsens = fopen("senswb.raw","wb");
                                        Fsein = fopen("seinwb.raw","wb");
//                                      cout << "Waarde vir integraal: ";
//                                      cin >> intgrwde;
                                        while (!donee) {
                                                while (intgrwde<2000 && !donee) {
                                                        ++intgrwde;
                                                        DAout(1,intgrwde);
                                                        fputc(lo(intgrwde),Fintg);
                                                        fputc(hi(intgrwde),Fintg);
                                                        TriggerAD();
                                                        wagsiklus(waitcount);
                                                        temp1 = GetSample();
                                                        fputc(lo(temp1),Fsein);
                                                        fputc(hi(temp1),Fsein);
                                                        fputc(lo(ISample),Fsens);
                                                        fputc(hi(ISample),Fsens);
```

```
                        if (kbhit())
                              donee = 1;
                  }

                  while (intgrwde>-2000 && !donee) {
                        --intgrwde;
                        DAout(1,intgrwde);
                        fputc(lo(intgrwde),Fintg);
                        fputc(hi(intgrwde),Fintg);
                        TriggerAD();
                        wagsiklus(waitcount);
                        temp1 = GetSample();
                        fputc(lo(temp1),Fsein);
                        fputc(hi(temp1),Fsein);
                        fputc(lo(ISample),Fsens);
                        fputc(hi(ISample),Fsens);
                        if (kbhit())
                              donee = 1;
                  }
            }
            fclose(Fintg);
            fclose(Fsens);
            fclose(Fsein);
      }
      break;
      case CKEY:
      {
//          clrscr();
            gotoxy(60,2);
            cout << "              ";
            gotoxy(60,2);
            cout << rotpos << " " << transpos << " ";
            cout << totpos;
      }
      break;
      case HKEY:
            print_mainmenu();
      break;
      case SKEY:
      {
            unsigned int uitwaarde;
            cout << "Sleutel D/A waarde in (-2047 < x < 2047): ";
            cin >> uitwaarde;
            DAout(0,0);
            DAout(1,uitwaarde);
      }
      break;
      case RKEY:
      {
            FILE *Fsens;
            FILE *Fsein;
            int done_r = 0, temp1 = 0;

            char senstr[16];
            char sigstr[16];
            cout << "Sensorleernaam: ";
            cin >> senstr;
            cout << "Seinleernaam: ";
            cin >> sigstr;

            Fsens = fopen(senstr,"wb");
            Fsein = fopen(sigstr,"wb");

            while (!done_r) {
                  if (kbhit())
                        done_r = 1;

                  TriggerAD();
```

```
                                        temp1 = GetSample();
                                        fputc(lo(temp1),Fsein);
                                        fputc(hi(temp1),Fsein);
                                        fputc(lo(ISample),Fsens);
                                        fputc(hi(ISample),Fsens);

                                    }
                                    fclose(Fsens);
                                    fclose(Fsein);
                                }
                                break;
                            }       //switch
                    }       //if

            }       // while

    }       // main
```

Bylae I – Programkode van die handoudiorestourasie-program vir Matlab

```
function varargout = hoofwin(varargin)
% HOOFWIN M-file for hoofwin.fig
%      HOOFWIN, by itself, creates a new HOOFWIN or raises the existing
%      singleton*.
%
%      H = HOOFWIN returns the handle to a new HOOFWIN or the handle to
%      the existing singleton*.
%
%      HOOFWIN('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in HOOFWIN.M with the given input arguments.
%
%      HOOFWIN('Property','Value',...) creates a new HOOFWIN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before hoofwin_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to hoofwin_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help hoofwin

% Last Modified by GUIDE v2.5 07-May-2003 11:11:39

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @hoofwin_OpeningFcn, ...
                   'gui_OutputFcn',  @hoofwin_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before hoofwin is made visible.
function hoofwin_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to hoofwin (see VARARGIN)

global LAST_S_BEG
LAST_BEG = 0;
% Choose default command line output for hoofwin
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% This sets up the initial plot - only do when we are invisible
% so window can get raised using hoofwin.
%if strcmp(get(hObject,'Visible'),'off')
%    plot(randn(1,10000));
%end

% UIWAIT makes hoofwin wait for user response (see UIRESUME)
```

```
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = hoofwin_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes1);
cla;

if ~(exist(get(handles.edit5,'String'),'file') == 2)
       set(handles.edit5,'String','Enter existing .wav file name')
else
       y = wavread(get(handles.edit5,'String'));
       plot(y);
end

aa = axis;

Update_Edit_Values(handles);

% --------------------------------------------------------------------
function FileMenu_Callback(hObject, eventdata, handles)
% hObject    handle to FileMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function OpenMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to OpenMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = uigetfile('*.fig');
if ~isequal(file, 0)
    open(file);
end

% --------------------------------------------------------------------
function PrintMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to PrintMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
printdlg(handles.figure1)

% --------------------------------------------------------------------
function CloseMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end
```

```
delete(handles.figure1)



% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double



% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

aa = axis;
FoW = aa(2)-aa(1)+1;

slidpos = get(hObject,'Value');
stepsize            =           get(hObject,'SliderStep');            %FoW            /
(2*(length(get(get(handles.axes1,'Children'),'YData'))-FoW) );
raamtal = slidpos/stepsize(1);
aa(1) = raamtal*(FoW/2)+1;
aa(2) = raamtal*(FoW/2)+FoW;
axis(aa);

Update_Edit_Values(handles);
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

aa = axis;
aa([3 4]) = 0.75*aa([3 4]);
axis(aa);
Update_Edit_Values(handles);

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

aa = axis;
aa([3 4]) = 1.25*aa([3 4]);
axis(aa);
Update_Edit_Values(handles);

% --- Update Axis Edit Boxes Values
function Update_Edit_Values(handles)
aa = axis;
set(handles.edit1,'String',num2str(aa(1)))
set(handles.edit2,'String',num2str(aa(2)));
set(handles.edit3,'String',num2str(aa(3)));
set(handles.edit4,'String',num2str(aa(4)));


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

axis tight
Update_Edit_Values(handles);


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

aa(1,1) = str2double(get(handles.edit1,'String'));
aa(1,2) = str2double(get(handles.edit2,'String'));
aa(1,3) = str2double(get(handles.edit3,'String'));
aa(1,4) = str2double(get(handles.edit4,'String'));
axis(aa)

FoW = aa(2)-aa(1)+1;
stepsize = FoW / (2*(length(get(get(handles.axes1,'Children'),'YData'))-FoW) ) ;
set(handles.slider2,'SliderStep',[stepsize 0.1]);

raamtal = (aa(1)-1)./(FoW/2);
slidpos = stepsize*raamtal;
set(handles.slider2,'Value',slidpos);



% --- Executes on button press in togglebutton2.
function togglebutton2_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton2

if ~isempty( findstr( 'on' , get(hObject,'String') ) )
      zoom off;
      set(hObject,'String','Zoom is off');
else
      zoom on
      set(hObject,'String','Zoom is on');
end


% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[x,y] = ginput(2);
x = sort(round(x));

set(handles.edit6,'String',num2str(x(1)));
set(handles.edit7,'String',num2str(x(2)));
set(handles.edit14,'String',num2str(x(2)-x(1)));


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

aa = round(axis);
sndclip = get(get(handles.axes1,'Children'),'YData');

if aa(1) < 1
      aa(1) = 1;
end
if aa(2) > length(sndclip)
      aa(2) = length(sndclip);
end
soundsc(sndclip(aa(1):aa(2)),str2double(get(handles.edit8,'String')))


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double



% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global S_BEG
global S_END
global ORIGINAL

sndclip = get(get(handles.axes1,'Children'),'YData');
sndclip(S_BEG:S_END) = ORIGINAL;
set(get(handles.axes1,'Children'),'YData',sndclip);


% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if findstr('file name',get(handles.edit10,'String'))
       set(handles.edit10,'String','Enter destination file name')
else
       sndclip = get(get(handles.axes1,'Children'),'YData');
       Fs = str2double(get(handles.edit8,'String'));
       wavwrite(sndclip./max(abs(sndclip)),Fs,get(handles.edit10,'String'));
end


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit12 as a double


% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global S_BEG;
global LAST_S_BEG;
global S_END;
global ORIGINAL;

S_BEG = str2double(get(handles.edit6,'String'));
S_END = str2double(get(handles.edit7,'String'));
G = str2double(get(handles.edit11,'String'));
iterasies = str2double(get(handles.edit13,'String'));
orde = str2double(get(handles.edit9,'String'));
corrlen = str2double(get(handles.edit15,'String'));

if LAST_S_BEG == S_BEG
       sndclip = get(get(handles.axes1,'Children'),'YData');
       to_rest = [sndclip(S_BEG-corrlen:S_END+corrlen)];
       sndclip(S_BEG:S_END)    =    G.*veldhuis(to_rest,middelblok(length(to_rest),S_END-
S_BEG+1),orde,iterasies);
       set(get(handles.axes1,'Children'),'YData',sndclip);
else
       sndclip = get(get(handles.axes1,'Children'),'YData');
       ORIGINAL = sndclip(S_BEG:S_END);
       to_rest = [sndclip(S_BEG-corrlen:S_END+corrlen)];
       sndclip(S_BEG:S_END)    =    G.*veldhuis(to_rest,middelblok(length(to_rest),S_END-
S_BEG+1),orde,iterasies);
       set(get(handles.axes1,'Children'),'YData',sndclip);
end

LAST_S_BEG = S_BEG;


% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a double


% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a double


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a double
```

```
%-----------------------------------------------------------------
function xhat = veldhuis(data,t,orde,iterasies)

% veldhuis.m
% function xhat = veldhuis(data,t,orde,iterasies)
% data - data frame
% t - indexpositions of corrupted data points
% orde - order van AR-process
% iterasies - number if iterations; iterasies=5 should do


v = data;
v(t) = 0;

a = real(lpc(v,orde));
b = do_b(a);
G_tilde = do_G_tilde(b,t,orde);
z = do_z(b,v,t);
xhat = G_tilde \ -z;
data(t) = xhat;

for ii = 1:iterasies-1
       a = real(lpc(data,orde));
       b = do_b(a);
       G_tilde = do_G_tilde(b,t,orde);
       z = do_z(b,v,t);
       xhat = G_tilde \ -z;
       data(t) = xhat;
end

%-----------------------------------------------------------------

function t = middelblok(dlen,m)

% middelblok.m
% function t = middelblok(dlen,m)

beg = (dlen-m)/2;
t = beg+1:beg+m;

%-----------------------------------------------------------------

function b = do_b(a);

% Kry vector 'b' vanuit parameters 'a'. Verwys na 'Adaptive Restoration of
% Unknown Samples in Discrete-Time Signals and Digital Images' deur R.N.J
% Veldhuis, 1988, p34.

b = xcorr(a);

%-----------------------------------------------------------------

function G_tilde = do_G_tilde(b,t,orde)

% do_G_tilde.m
% Kry matriks 'G_tilde' vanuit parameters 'b' & 't'. Verwys na 'Adaptive Restoration of
% Unknown Samples in Discrete-Time Signals and Digital Images' deur R.N.J
% Veldhuis, 1988, p36.


% indien onbekende monsters almal opeenvolgend sou wees:
% if length(t) > beg
%      b = [b(beg:end) zeros(1,length(t)-beg)];
% else
%      b = b(beg:beg+length(t)-1);
% end
%
% G_tilde = toeplitz(b);
```

```
G_tilde = zeros(length(t));
for ii = 1:length(t)
      for jj = 1:length(t)
            if ((t(jj)-t(ii)+orde+1) > length(b)) | ((t(jj)-t(ii)+orde+1) <= 0)
                  G_tilde(ii,jj) = 0;
            else
                  G_tilde(ii,jj) = b(t(jj)-t(ii)+orde+1);
            end
      end
end

%-----------------------------------------------------------------

function z = do_z(b,v,t)

% do_z.m
% Kry vektor 'z' vanuit parameters 'b', 'v' & 't'. Verwys na 'Adaptive Restoration of
% Unknown Samples in Discrete-Time Signals and Digital Images' deur R.N.J
% Veldhuis, 1988, p36. (Daar is 'n fout in Veldhuis se vergelyking!!)

orde = (length(b)-1)/2;
if size(v,1) ~= 1
      v = v';
end

% Maak 'n matriks van vektor 'v'.
V = zeros(length(t),length(b));
for ii = 1:length(t)
      V(ii,:) = v(t(ii)-orde:t(ii)+orde);
end

z = V * b';
```

Verwysings

[1] T. Kessler and S. Ziegler, "Direct playback of negatives of historic sound cylinders," *EVA Europe '99*, pp. 8-1 - 8-5, Berlin, Germany, 1999.

[2] T. Asakura, T. Iwai, et al., "Reproduction of the Sounds from old wax phonographic cylinders using the laser-beam reflection method," *Proc. IEEE ICASSP '86*, pp. 493-496, Tokyo, Japan, 1986.

[3] T. Nakamura, "Laser positioning," *The Phonograph Makers' Pages*, http://members01.chello.se/christer.hamp/phono/nakamura.html, September 2002

[4] J. Poliak, "The Optical Turntable," *The Phonograph Makers' Pages*, http://members01.chello.se/christer.hamp/phono/poliak.html, June 2002

[5] V. V. Petrov, "The Digital Optomechanical Method," *The Phonograph Makers' Pages*, http://members01.chello.se/christer.hamp/phono/petrov.html, July 2002

[6] V. V. Petrov, S. M. Shanoylo, et al., "Optomechanical method of Edison cylinders sound reproduction," *An Audio Eng. Soc. Preprint (Preprint 4491), Munich*, 1997.

[7] D. C. Giancoli, *Physics*, Fifth ed., New Jersey: Prentice-Hall, 1998.

[8] K. J. Kuhn, "Audio Compact Disk - An Introduction," http://www.ee.washington.edu/conselec/CE/kuhn/cdaudio/95x6.htm, June 14 2001

[9] K. C. Pohlmann, *Priciples of digital audio*, Third ed.: McGraw-Hill, Inc., 1995.

[10] "Sam's laser FAQ," http://www.laserfaq.com/laserdio.htm, May 2001

[11] G. F. Franklin, J. D. Powell and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, Third ed.: Addison-Wesley, Inc., 1994.

[12] G. F. Franklin, J. D. Powell and M. Workman, *Digital Control of Dynamic Systems*, Third ed., Menlo Park, Ca.: Addison Wesley Longman, Inc., 1998.

[13] J. C. Russ, *The Image Processing Handbook*, Second ed., Boca Raton, Fla.: CRC Press, 1995.

[14] R. N. Veldhuis, "Adaptive Restoration of Unknown Samples in Diskrete-Time Signals and Digital Images," Ph.D Thesis, Katholieke Universiteit te Nijmegen, 1988

[15] S. J. Godsill, P. J. W. Rayner and O. Cappé, "Digital audio restoration," in M. Kahrs and K. Brandenburg, Eds., *Applications of Digital Signal Processing to Audio and Acoustics*, pp. 133-193, Kluwer Academic Publishers, 1998.

[16] O. Cappé, "Elimination of the Musical noise Phenomenon with the Ephraim and Malah Noise Suppressor," *IEEE Trans. on speech and audio processing*, vol. 2, pp. 345-349, 1994.

[17] P. J. Wolfe and S. J. Godsill, "Towards a perceptually optimal spectral amplitude estimator for audio signal enhancement," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 821-824, Istanbul, Turkey, 2000.

[18] S. M. Kay, *Modern Spectral Estimation: Theory & Application*, Englewood Cliffs, NJ: Prentice-Hall, 1988.

[19] Horowitz and Hill, *The art of electronics*, Low price ed., Cambridge: University Press, 1995.

[20] "The Phonograph Makers' Pages," http://members01.chello.se/christer.hamp/phono/, September 2002

[21] D. C. Lay, *Linear algebra and its applications*, Second ed.: Addison-Wesley, 1997.

[22] T. Nakamura, "Optical Reproduction of Sounds from Old Wax Phonograph Cylinders," *Kushiro National College of Technology*, http://w3.kushiro-ct.ac.jp/hikari/waxcyl/waxcyl_e.html, September 2002

[23] J. W. Nilsson and S. A. Riedel, *Electric Circuits*, Fifth ed.: Addison-Wesley, Inc., 1996.

[24] V. V. Petrov, A. Kryuchin, et al., "Conservation and Introduction into the Scientific Circulation of Rarity Musical Collections," http://www.evarussia.ru/upload/doklad/dokladEn_307.doc, September 2002

[25] G. D. Forney, "The Viterbi Algorithm," *Proc. IEEE*, vol. 61, pp. 268 - 278, 1973.

[26] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*, Third ed., New Jersey: Prentice-Hall, Inc., 1996.

[27] L. R. Rabiner and B. H. Juang, "An Introduction to Hidden Markov Models," in *IEEE ASSP Magazine*, pp. 4 - 16, January 1986

[28] G. Sharpless, "Introduction to CD and CD-ROM," http://www.disctronics.co.uk/downloads/tech_docs/cdintroduction.pdf, June 2002

[29] G. Stanke, "Inside Out," *The Phonograph Makers' Pages*, http://members01.chello.se/christer.hamp/phono/stanke.html, June 2003