Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Automatisering og signalbehandling	Vårsemesteret, 2017 Åpen / ~~Konfidensiell~~
Forfatter: Øyvind Stensland	Ø. Stensland (signatur forfatter)
Fagansvarlig: Kjersti Engan Veileder(e): Kjersti Engan	
Tittel på masteroppgaven: Deteksjon av kompresjoner ved bruk av lydopptak fra smarttelefon under trening på hjerte- lungeredning Engelsk tittel: Detection of compression using sound recording from smartphone during training on cardiopulmonary resuscitation.	
Studiepoeng: 30	
Emneord: Signalbehandling	Sidetall: 75 + vedlegg/annet: 112 Stavanger, 09.06.17 dato/år

Forord

Denne masteroppgaven utgjør den avsluttende delen av masterstudiene i Automatisering og Signalbehandling ved Universitetet i Stavanger.

Jeg vil takke min veileder Kjersti Engan for god oppfølging og veiledning gjennom hele prosjektet.

Jeg ønsker også å takke Tonje Soraas Birkenes, Helge Myklebust, og Solveig Haukås Haaland fra Laerdal Medical As for god hjelp underveis. Til slutt vil jeg rette en takk til familie, kamerater og ansatte på Laerdal for bidrag med å samle inn datasett til bruk i oppgaven.

Sammendrag

Trening på hjerte- lungeredning (HLR) utføres ofte i skolesammenheng med MiniAnne-dukker vist i figur 1.1a. Disse lager dobbel klikkelyd når det komprimeres dypt nok. Dersom elevene kan få en tilbakemelding på hvor bra HLR utføres, kan dette skape konkurranse blant elevene og følgelig bedre kompresjoner. Ideen er at hver elev bruker sin egen smarttelefon til å detektere korrekte kompresjoner for sin dukke. Dataene kan presenteres for alle elevene på en felles skjerm. Det blir også lettere for veileder å vite hvilke elever som trenger mest hjelp.

I rapporten er det utforsket mulighetene til å detektere klikkelyder fra MiniAnne med en smarttelefon. Datamateriellet som brukes i eksperimentene er samlet inn underveis ved hjelp av ansatte på Laerdal, familie og kamerater. Totalt 7 datasett ble samlet inn underveis.

Totalt tre forskjellige metoder ble testet for å detektere kompresjoner. *Template matching*, *blind source separation* og adaptiv amplitude terskling. I *template matching* konvolveres lydsignalet med en mal før kompresjonene detekteres med adaptiv terskling. I adaptiv amplitude terskling detekteres toppunkt over en terskelverdi. Dette brukes til å finne doble klikkelyder som er tilfelle for en kompresjon som er dyp nok. I *Blind source separation* skilles lyden fra hver dukke fra hverandre før det gjøres en adaptiv amplitude terskling.

Av de tre forskjellige metodene som ble testet gav *blind source separation* dårligst resultat. Det var vanskelig å skille lyden fra hver dukke fra hverandre. Bedre gikk det med *Template matching* hvor resultatene ble relativt gode. Her ble rundt 80% av kompresjonene ble detektert. Den enkleste metoden, adaptiv amplitude terskling gav best resultater hvor opp mot 100% av kompresjonene ble detektert.

Konklusjonen fra oppgaven er at det fungerer å detektere kompresjoner fra MiniAnne som er dype nok med en smarttelefon. En avstand mellom personer på over 1.2m i hver retning vil bli anbefalt for at kompresjonsdeteksjonen skal være tilstrekkelig robust.

Innhold

Liste over forkortelser

BSS Blind Source Separation

HLR Hjerte- lungeredning

ICA Independent Component Analysis

Kapittel 1

Innledning

1.1 Motivasjon

Hjertestans utenfor sykehus er en stor dødsårsak over hele verden. Hvert år rammes mellom 3000 og 5000 mennesker av plutselig hjertestans i Norge [1]. Av disse overlever mellom 10 og 20%. Til sammenligning dør under 150 mennesker i trafikkulykker i Norge [2]. Med tidlig hjerte- lungeredning (HLR) mer enn dobles sjansen til å overleve hjertestans [1]. I de senere år har det vært mye fokus på hjerte- lungeredning som har gitt gode resultater [3]. Stadig flere blir opplært til å utføre hjerte- lungeredning og overlevelsesprosenten går opp.

Gjennomsnittlig utrykningstid for ambulansetjenesten i Oslo ved hjertestans er på rundt fire minutter i sentrale strøk og opp til over ti minutter i utkanten av Oslo [4]. Enda lenger utrykningstid er det andre steder i Norge. Dette krever god utholdenhet til personer som utfører tidlig HLR. For at enda flere skal bli flinkere til å utføre HLR trengs det mer og riktig trening.

Riktig trening kan oppnås dersom de som trener på HLR får tilbakemelding på om treningen utføres riktig eller hva som skal til for å utføre det riktig. Konkurranse mellom elever på egen skole eller mellom skoler i en region kan bidra til å øke mengden HLR trening hver elev får.

1.2 Oppgavebeskrivelse

Når man trener på hjerte- lungeredning (HLR) gjøres dette ved hjelp av en treningsdukke. MiniAnne er en enkel oppblåsbar dukke som oftest brukes i HLR-opplæring i skolen. Denne dukken inneholder en metallplate som avgir en skarp klikkelyd når man komprimerer dypt nok, se figur 1.1b. Lyden fra klikkeren skal tas opp med en smarttelefon og brukes til å telle antall kompresjoner som er dype nok samt kompresjonsrate. En passenede signalbehandlings-algoritme må utvikles for å detektere klikkelyder. Dermed kan man finne ut hvor mange gode kompresjoner hver elev har utført og hvor bra HLR er utført. Dette kan vises til elevene for å motivere til å gjøre en best mulig innsats under HLR opplæring. Dataene kan vises til lærerne og til organisasjonen som tilbyr læreprogrammet. I tillegg kan dataene sammenlignes med andre skoler for å skape konkurranse. Hensikten er å stimulere elever og lærere til å ha størst mulig fokus på gode kompresjoner og på mengdetrening. Oppgaven er utført i samarbeid med Laerdal Medical.

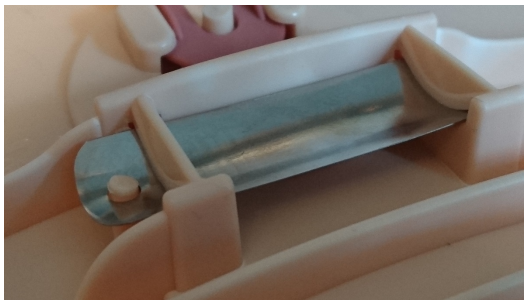
1.3 Bakgrunn

Dersom en person får hjertestans er tidlig hjerte- lungeredning (HLR) avgjørende for om personen overlever eller ikke [3]. Det anbefales å utføre 30 kompresjoner og 2 innblåsninger. Kompresjonene bør gjøres i en hastighet av 100 til 120 pr. minutt [5]. Dette er tatt hensyn til når kompresjonene skal detekteres senere i rapporten. For mer info om HLR henvises det til Norsk Resuscitasjonsråd sine retningslinjer om HLR fra 2015 [5].

MiniAnne dukkene er billige oppblåsbare dukker som brukes når man trener på hjerte-lungeredning, se figur 1.1a. Dukkene har en metallplate som lager skarp klikkelyd når man komprimerer dypt nok, se figur 1.1b. Denne lager klikkelyd både på vei ned og på vei opp.

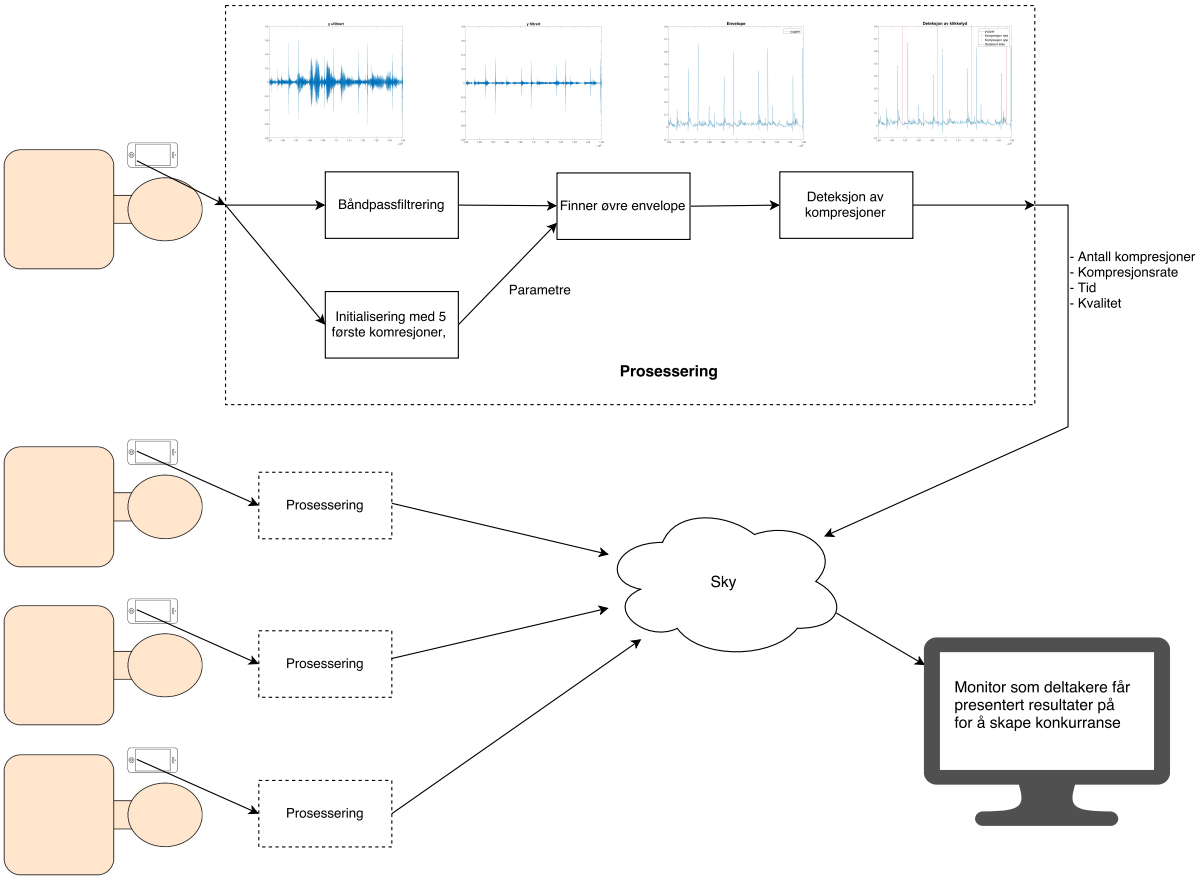
HLR trening med MiniAnne utføres i stor grad i skolesammenheng med en lærer uten god nok kompetanse om HLR som veileder. En skole mottar et sett med MiniAnne-dukker og en video som forklarer HLR. HLR-treningen i skolen er i stor grad passiv læring hvor elevene lærer ved å følge videoen. Man vet verken hvor lenge hver elev trener eller hvor godt HLR utføres.

(a) MiniAnne dukken



(b) Metallplaten som lager klikkelyd når man komprimerer dypt nok.

Dette vil Laerdal Medical gjøre noe med. Tanken er å bringe mer teknologi inn i HLR-trening for å motivere elever til å trene mer på HLR, og for å få tilbakemelding på hvor lenge hver elev trener og hvor godt HLR utføres. Et overordnet system som kan brukes er vist i figur 1.2. Her legges en smarttelefon foran dukken som tar opp lyd og detekterer kompresjoner som er dype nok. Hver elev får tilbakemelding på smarttelefonen hvor bra kompresjonen utføres. I tillegg kan data så sendes til en server og presenteres på en skjerm for elevene. På den måten blir det konkurranse mellom elever som øker motivasjon og mengden trening. Skoler kan konkurrere med skoler og regioner med regioner. Denne rapporten vil kun ta for seg utvikling av algoritmen markert med stiplet linje i figur 1.2.



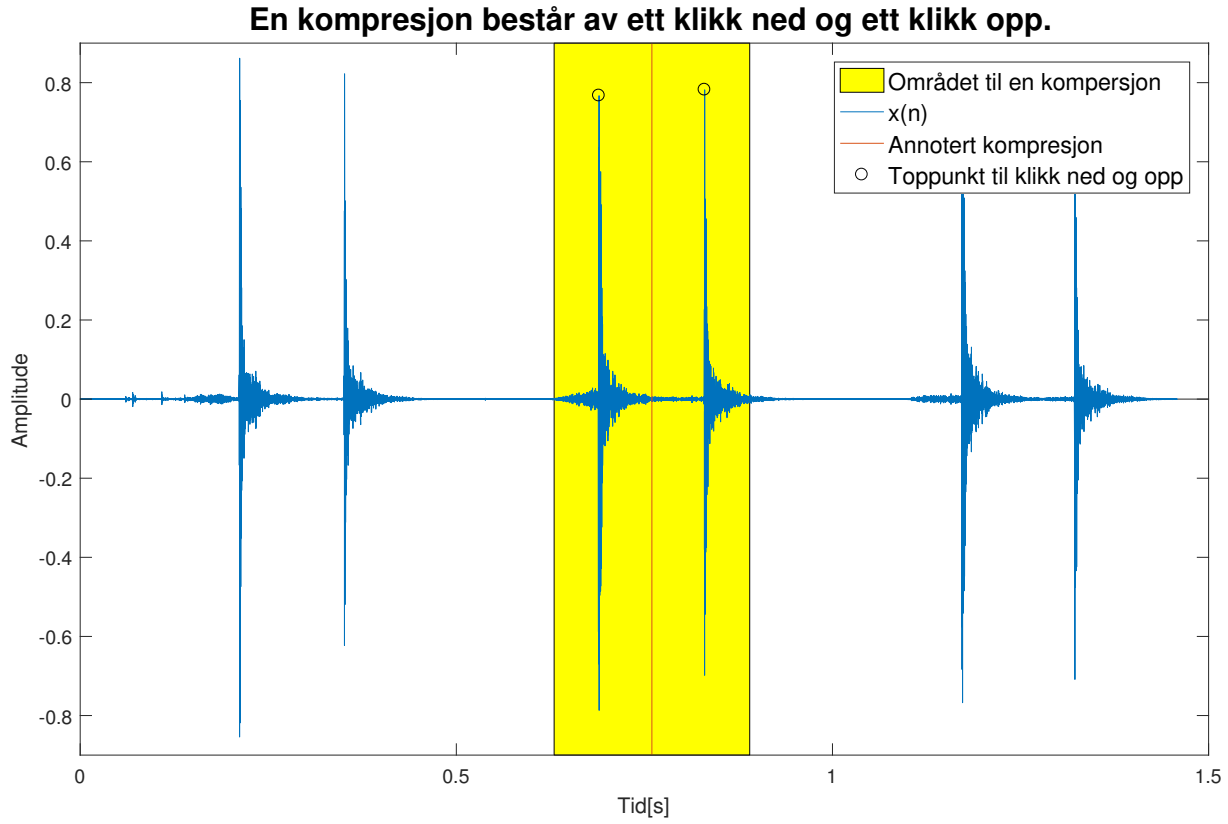
Figur 1.2: Overordnet oversikt over mulig system.

Det er i hovedsak vurdert to forskjellige metoder dette problemet kan løses på:

- Kun sanntid tilbakemelding på hvor gode kompresjonene er. Dataene kan i tillegg sendes til en server slik at de kan presenteres på en skjerm for å skape konkurranse blant elevene. Dette tilsvarer metode A og C fra kapittel 4.2 og 4.4.
- Ingen sanntid tilbakemelding. All data blir sendt til en server eller enhet hvor prosesseringen foregår. Dataene kan i ettertid presenteres for elevene. Dette tilsvarer metode B fra kapittel 4.3.

Det vil være en stor fordel å kunne gi sanntid tilbakemelding slik at personen som trener på HLR kan forbedre seg underveis. Tilbakemelding i ettertid vil i liten grad hjelpe personer som trener på HLR men kan brukes til å finne ut hvor dyktige personer er til å utføre HLR.

En kompresjon består av ett klikk ned og ett klikk opp på klikkeren i figur 1.1b. En kompresjon er markert i gult i figur 1.3. For å kunne detektere denne må man finne to klikk etter hverandre med en viss avstand. I figur 1.3 er det kun en som komprimerer. Ikke alle lydopptak gir like spisse klikk som figur 1.3 viser. I figur 5.9 er det fem personer som komprimerer samtidig. Dette lydopptaket er i tillegg tatt opp med en telefon som har ekstra sensitiv mikrofon, noe som fører til at lydopptaket ble støyfullt. Dette må tas hensyn til når algoritmen for å detektere kompresjoner utvikles.



Figur 1.3: En kompresjon består av to klikk, ett ned og ett opp. Her markert med gult. De sorte sirklene markerer toppunkt på klikk ned og klikk opp. Midt mellom sirklene er det annotert en kompresjon, $Det(i)$ markert med vertikal rød linje.

1.4 Notasjon og variabelnavn

Gjennom hele oppgaven er det brukt *kursiv* tekst for engelske ord som det ikke er beskrivende nok norske ord for. Vektorer er skrevet med små bokstaver og matriser med store bokstaver. Notasjonen som er brukt i rapporten er som følger:

- $x(n)$: Originalt ubehandlet lydopptak
- $x_{bp}(n)$: Båndpassfiltrert $x(n)$
- $x_{env}(n)$: Øvre envelope til $x_{bp}(n)$
- $x_{conv}(n)$: $x_{env}(n)$ konvolvert med template
- $Det(i)$: Tidspunkt for detektert kompresjon beståene to klikk, se figur 1.3. Tidspunktet vil da være midt mellom tidspunktet for klikk 1 og 2. Under *template matching* er det kun ett toppunkt for hver kompresjon, se figur 2.4.
- $Det_{amp}(i)$: Amplituden til detektert kompresjon $Det(i)$. I metode B og C vil det være to toppunkt (klikk ned og klikk opp) som definerer en kompresjon. Da blir $Det_{amp}(i)$ gjennomsnittlig amplitude på klikk ned og klikk opp.

Tabell 1.1 viser alle parametre som er nevnt i denne rapporten og som er brukt i algoritmene som blir beskrevet. Kapittel 5.4 forklarer valg av verdi for hver parameter som har konstant verdi fra tabell 1.1. Når parametre blir nevnt i rapporten er de skrevet omgitt med tøddler “”.

Variabelnavn	**Verdi**	**Informasjon**
"dobbelKlikkDistanse"	19 200 sampler (400ms)	Minste mulig avstand mellom to kompresjoner, tilsvarer en kompresjonsrate på 150pr minutt
"enkelKlikkDistanseMin"	3 000 sampler, (62,5ms)	Angir minste mulige avstand mellom to enkle klikk for at det skal kunne bli akseptert som en kompresjon
"enkelKlikkDistanseMaks"	12 000 sampler, (250ms)	Angir største mulige avstand mellom to enkle klikk for at det skal kunne bli akseptert som en kompresjon
"maksVerdi"	adaptiv	Følger gjennomsnittet av amplituden til klikk ned og klikk opp
"terskelFaktor"	0.3, (0.6 i metoden *Template Matching*)	Forholdstall mellom "maksVerdi" og "terskelVerdi"
"terskelVerdi"	"terskelFaktor" * "maksVerdi"	Følger "maksVerdi", alle klikk tilhørende riktig dukke vil ha amplitude høyere enn dette.
"toppunktMin"	adaptiv	Amplituden til toppunkt med lavest amplitude i en kompresjon som består av to toppunkt. Brukes under kompresjonsdeteksjon
"absoluttMaks"	adaptiv	Høyeste amplitude på hele lydopptak til nå. Endres dersom høyere amplituder forligger.

Tabell 1.1: Liste over variabler brukt i denne rapporten

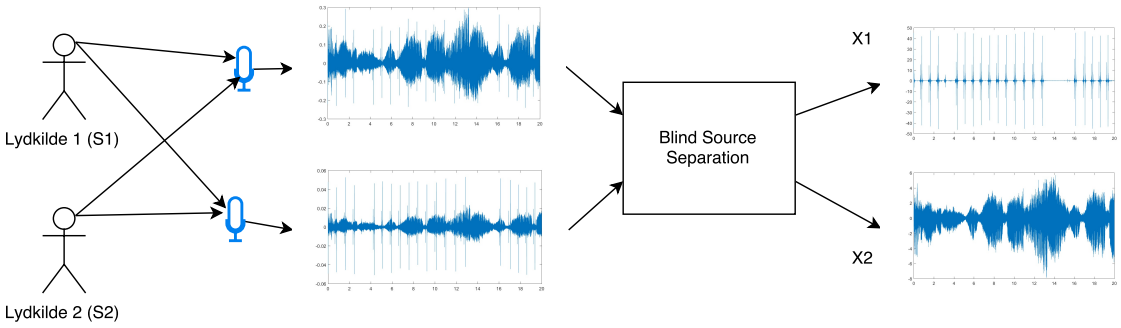
Kapittel 2

Teori

Dette kapittelet vil forklare teorien som er brukt videre i rapporten. Dette innebærer *Blind Source Separation*, *Template Matching* og *Envelope*.

2.1 Blind Source Separation

I et rom hvor flere mennesker snakker sammen er det relativt lett for mennesker å følge med på en samtale og skille ut andre lyder som bakgrunnsstøy. Dersom man tar lydopptak med flere mikrofoner i et slikt rom hvor flere mennesker snakker sammen (lydkilder), blir lydkildene mikset sammen forskjellig i hver mikrofon. Det å kunne skille en lydkilde fra en annen er ikke helt rett frem for en datamaskin. Lydkildene blir mikset sammen, i tillegg reflekteres lyder i vegger og man får med reflekterte versjoner av lydkildene. Problemet med å skille en lyd fra en annen ved bruk av datamaskin ble først definert i 1953 av E. Colin Cherry som the "cocktail party problem" [6]. Dette er et vanlig eksempel på *Blind Source Separation* (BSS) hvor man har flere lydkilder og skal trekke ut én lydkilde fra flere mikser av lydkildene. Figur 2.1 viser et eksempel hvor det er to lydkilder og to mikrofoner. Lydkildene mikses sammen i mikrofonene og separeres siden ved hjelp av BSS. For å kunne separere lydkildene er man nødt til å vite hvordan lydene har blitt mikset.



Figur 2.1: Eksempel på BSS hvor to lydkilder mikses og siden separeres med BSS.

Gitt eksempelet fra figur 2.1 hvor det er to lydkilder, s_1 og s_2, blir tatt opp med to mikrofoner, x_1 og x_2. Signalet fra mikrofonene kan da modelleres som et lineært system [7]:

$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t) \tag{2.1}$$

$$x_2(t) = a_{21}s_1(t) + a_{22}s_2(t) \tag{2.2}$$

hvor a_{ij} er impuls-responsen til rommet. Denne vil være variere med avstand til kildesignal, og hvordan rommet reflekterer lyder. Disse to ligningene kan skrives på matriseform:

$$\mathbf{X} = \mathbf{AS} \tag{2.3}$$

hvor

$$\mathbf{X} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \mathbf{S} = \begin{bmatrix} s_1(t) \\ s_2(t) \end{bmatrix} \tag{2.4}$$

For å finne tilbake til lydkilden s må man dekonvolvere $\mathbf{X}$ med impuls-responsen til rommet [7]:

$$\mathbf{U} = \mathbf{WX} \tag{2.5}$$

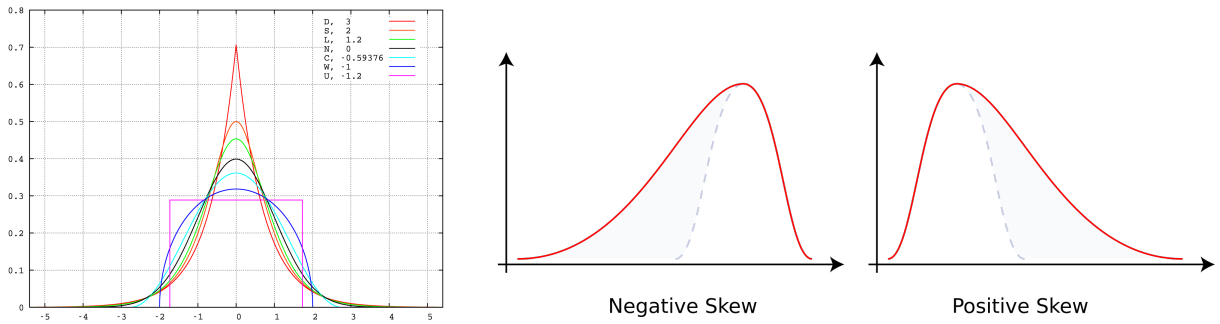
hvor $\mathbf{W} = \mathbf{A}^{-1}$ og $\mathbf{U}$ er et estimat av original lydkilde $\mathbf{S}$. Problemet med å finne $\mathbf{W}$ kalles *Blind Source Separation* fordi $\mathbf{W}$ er ukjent og må estimeres. En metode å finne $\mathbf{W}$ på er ved bruk av *Independent Component Analysis* (ICA).

2.1.1 Independent Component Analysis

Det finnes flere algoritmer [8] for å utføre *Independent Component Analysis* (ICA). FastICA som er laget ved universitetet i Helsinki [9] er regnet som den mest effektive og vil være utgangspunktet for denne beskrivelsen av ICA. ICA er en statistisk metode for å finne skjult informasjon i et sett med tilfeldige variable [10]. Dette kan for eksempel være lydkilder i et sett miksede lydsignaler som i figur 2.1. Ved bruk av ICA forutsettes det at kildesignalene er uavhengig og ikke gaussisk fordelt [11]. Ved miksing av kildesignalene **S** som gjøres i mikrofonen ser man følgende [12]:

- De miksede signalene **X** er avhengig av hverandre selv om kildesignalene **S** ikke er avhengig av hverandre.
- Sentralgrenseteoremet tilsier at i de fleste tilfeller, når uavhengige variable legges sammen, går summen mot en normalfordeling, selv om de opprinnelige variablene ikke er normalfordelt [13]. **X** er dermed mer gaussisk fordelt enn **S**.
- Kompleksiteten til et mikset signal **X** er er større enn kompleksiteten til et kildesignal **S**.

En måte å utføre ICA på er ved å endre **W** helt til **U** er minst mulig gaussisk, kompleks og avhengig. For å finne ut hvor gaussisk signalet er, er det vanlig å benytte kurtose (spisshet). Kurtose er fjerde moment statistikk (gjennomsnitt, varians og skjevhet er de tre første) som viser hvor spiss tetthetsfunksjonen er, se figur 2.2a.



(a) Spisshet (kurtosis)

(b) Skjevhet (skewness)

Figur 2.2: Figuren viser spisshet og skjevhet

Kurtose er vanligvis definert som [9]:

$$K(y) = E\{y^4\} - 3(E\{y^2\})^2 \tag{2.6}$$

Negativ entropi (negentropi) er en annen metode for å finne ut hvor gaussisk et signal er. Negativ entropi er et mål på hvor sterkt en stokastisk variabel avviker fra en uniform fordeling [14]. Entropien H av en tilfeldig variabel Y er definert som:

$$H(y) = -\sum_i P(Y = a_i) log(P(Y = a_i)) \tag{2.7}$$

hvor a_i er de mulige verdiene Y kan ha. Av alle fordelinger med gitt gjennomsnitt og varians er det normalfordelingen som har høyest entropi [15]. Dermed kan entropi brukes som et mål på hvor gaussisk en variabel er. For å få et tall som er null for en gaussisk variabel og som aldri blir negativ blir negentropi brukt. Negentropi J er definert som:

$$J(y) = H(y_{gauss}) - H(y) \tag{2.8}$$

hvor $H(y)$ entropien til en tilfeldig variabel y og $H(y_{gauss})$ er entropien til en gaussisk tilfeldig variabel med samme varians og gjennomsnitt som y. Definisjonen av negentropi i ligning 2.8 er vanskelig å regne ut og blir derfor tilnærmet med:

$$J(Y) \approx \sum_i k_i[E(G_i(y)) - E(G_i(y_{gauss}))]^2 \tag{2.9}$$

hvor k er positive konstanter, y_{gauss} er normalfordelt variabel $\sim$ N(0,1), og y er antatt å ha gjennomsnitt 0 og varians lik 1. G_i er en ikke-kvadratisk funksjon. Denne funksjonen blir valgt slik at den ikke vokser for raskt. Følgende eksempler på G_i er anbefalt i FastICA algoritmen:

$$G_1(u) = \frac{1}{a_1} tanh(a_1 u), G_2(u) = u * e^{\frac{-u^2}{2}} \tag{2.10}$$

hvor $1 \leq a_1 \leq 2$ er brukbar konstant, ofte velges $a_1 = 1$. $\mathbf{W}$ oppdateres iterativt med følgende ligning:

$$w_{i,ny} = E\{xG(w_i^T x)\} - E\{G'(w_i^T x)\} w_i \tag{2.11}$$

Hvordan ICA utføres i praksis for å redusere kurtosen kan illustreres med et eksempel.

Gitt to uniformt fordelte variable s_1 og s_2 i en matrise $\mathbf{S}$ med 4000 punkter hver. Disse blir mikset med en matrise $\mathbf{A}$, se ligning 2.3. Figur 2.3a viser fordelingen til $\mathbf{X}$. Histogrammene på hver side av figuren viser tilnærmet tetthetsfunksjon. Før selve ICA utføres gjøres noe preprosessering. Første steg er å trekke gjennomsnittet fra $\mathbf{X}$. Dette er vist i figur 2.3b hvor fordelingen er sentret rundt 0. Neste steg er å dekorrelere signalene $\mathbf{X}$ som vist i figur 2.3c. Dette kan gjøres med en lineær transformasjon [9]:

$$\tilde{\mathbf{X}} = \mathbf{V}\mathbf{X} \tag{2.12}$$

hvor $\tilde{\mathbf{X}}$ er dekorrelert $\mathbf{X}$ og $\mathbf{V}$ er en transformasjonsmatrise. En vanlig måte å gjøre dekorrelering på er ved bruk av *eigen-value decomposition* (EVD) av kovarians-matrisen:

$$E\{\mathbf{X}\mathbf{X}^T\} = \mathbf{E}\mathbf{D}\mathbf{E}^T \tag{2.13}$$

hvor $\mathbf{E}$ er ortogonalmatrise av egenvektorer av $E\{\mathbf{X}\mathbf{X}^T\}$ og $\mathbf{D}$ er diagonal matrise av egenverdier. Dermed kan dekorrelering gjøres med

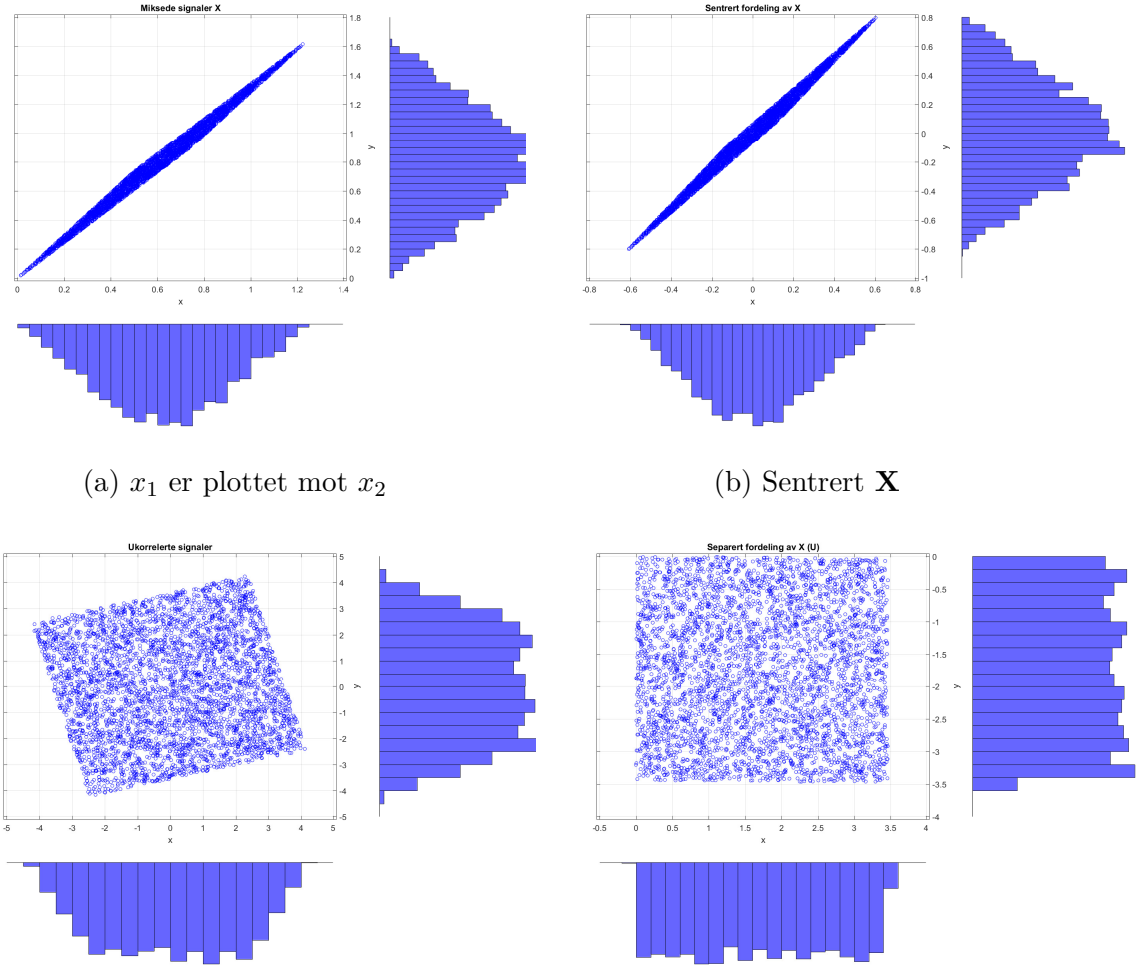
$$\tilde{\mathbf{X}} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{X} \tag{2.14}$$

Når all preprosesseringen er ferdig kan separerte signaler finnes ved en ortogonal transformasjon av $\tilde{\mathbf{X}}$. Dette er i praksis en rotasjon av felles sannsynlighetsfordeling slik som vist i figur 2.3d. FastICA algoritmen regner ut en og en vekt w_i i $\mathbf{W}$ matrisen for å finne optimal vekt for hver uavhengig komponent, $w_i^T x$. De iterative stegene er som følger:

1. Velg en tilfeldig vekt w_i
2. $w_{i,ny} = E\{xG(w_i^T x)\} - E\{G'(w_i^T x)\}w_i$
3. $w_i = \frac{w_{i,ny}}{|w_{i,ny}|}$
4. Dersom $w_{i,ny}$ ikke har konvergert mot w_i gå tilbake til steg 2.

Konvergens oppnås når to $w_{i,ny}$ og w_i peker i samme retning, da er prikkproduktet tilnærmet lik 1. Etter hver iterasjon blir $\mathbf{W}$ dekorrelert for å hindre at to w_i konvergerer mot samme punkt. $\mathbf{W}$ blir så multiplisert med $\mathbf{X}$ for å finne et estimat av kildesignalene som ligning 2.5 viser.

Den riktige rotasjonen er den som gir lavest mulig kurtose, eller høyest mulig negentropi, av tetthetsfunksjonen, se figur 2.2a. Histogrammene i figur 2.3 som er en tilnærming av tetthetsfunksjonen viser at kurtosen er lavest i figur 2.3d. På grunn av sentralgrenseteoremet vil miksen av signalene $\mathbf{X}$ ha mer gaussisk fordeling enn kildesignalene. Dermed er det i figur 2.3d mest sannsynlig funnet tilbake til et estimat av kildesignalene $\mathbf{S}$.



(a) x_1 er plottet mot x_2

(b) Sentrert $\mathbf{X}$

(c) Dekorrelert $\mathbf{X}$

(d) Separert $\mathbf{X}$, har nå funnet estimat av S

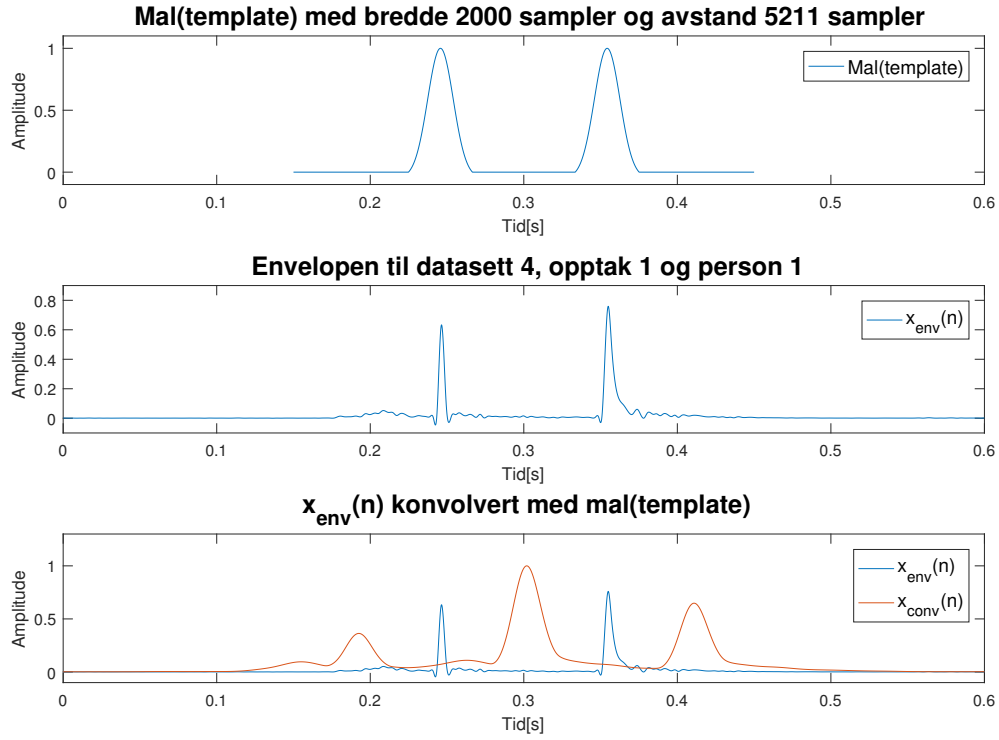
Figur 2.3: Figuren viser hvordan preprosessering og ICA fungerer.

2.2 Template matching

Template matching er en mye brukt teknikk i signalbehandling og blir blant annet brukt for å navigere robot [16], detektere lungeknuter i CT bilder [17] og til å detektere lydsignaler [18]. *Template matching* vil si å sammenligne et signal med en mal (*template*). Signalet kan være både 1D, 2D eller høyere dimensjon. En enkel måte å utføre *template matching* på er ved bruk av konvolusjon:

$$(f * g)(t) \stackrel{def}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.15}$$

hvor f er signalet og g er malen. Malen blir gradvis forskjøvet over signalet med tiden. Dermed får en høy verdi når malen stemmer overens med signalet. Figur 2.4 viser konvolusjon mellom et lydsignal f og en mal g. Her blir resultatet én høy topp hvor malen stemmer overens med lydsignalet.



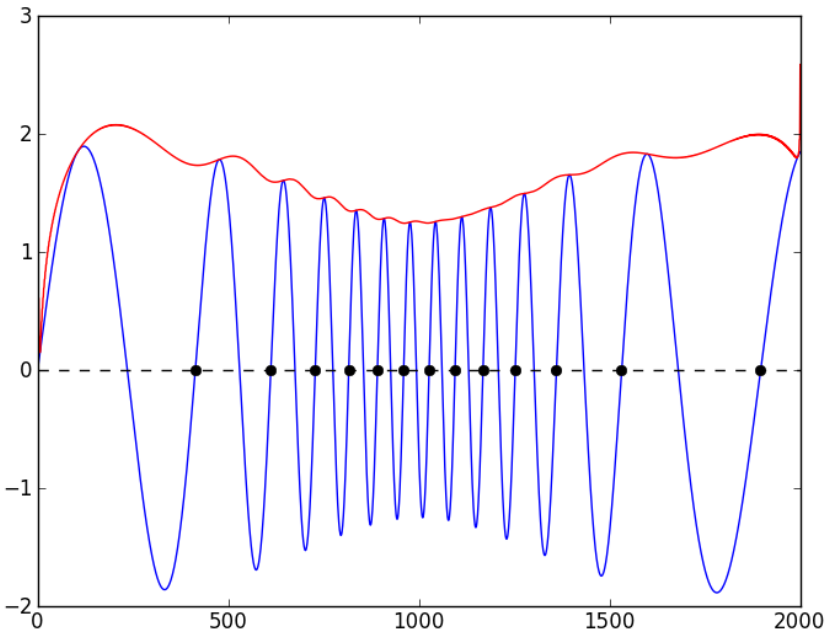
Figur 2.4: Konvolusjon mellom et lydsignal f og en mal g.

2.3 Envelope

Envelopen til et signal, eller konvolutt som det heter direkte oversatt til norsk, er en glatt kurve som beskriver ekstremalpunktene til signalet [19], se figur 2.5. *Envelopen* kan finnes på minst tre forskjellige måter:

- Finn alle toppunkt eller bunnpunkt til signalet og interpoler mellom disse.
- Skyv et vindu over signalet.
- Ved bruk av Hilbert Huang filter.

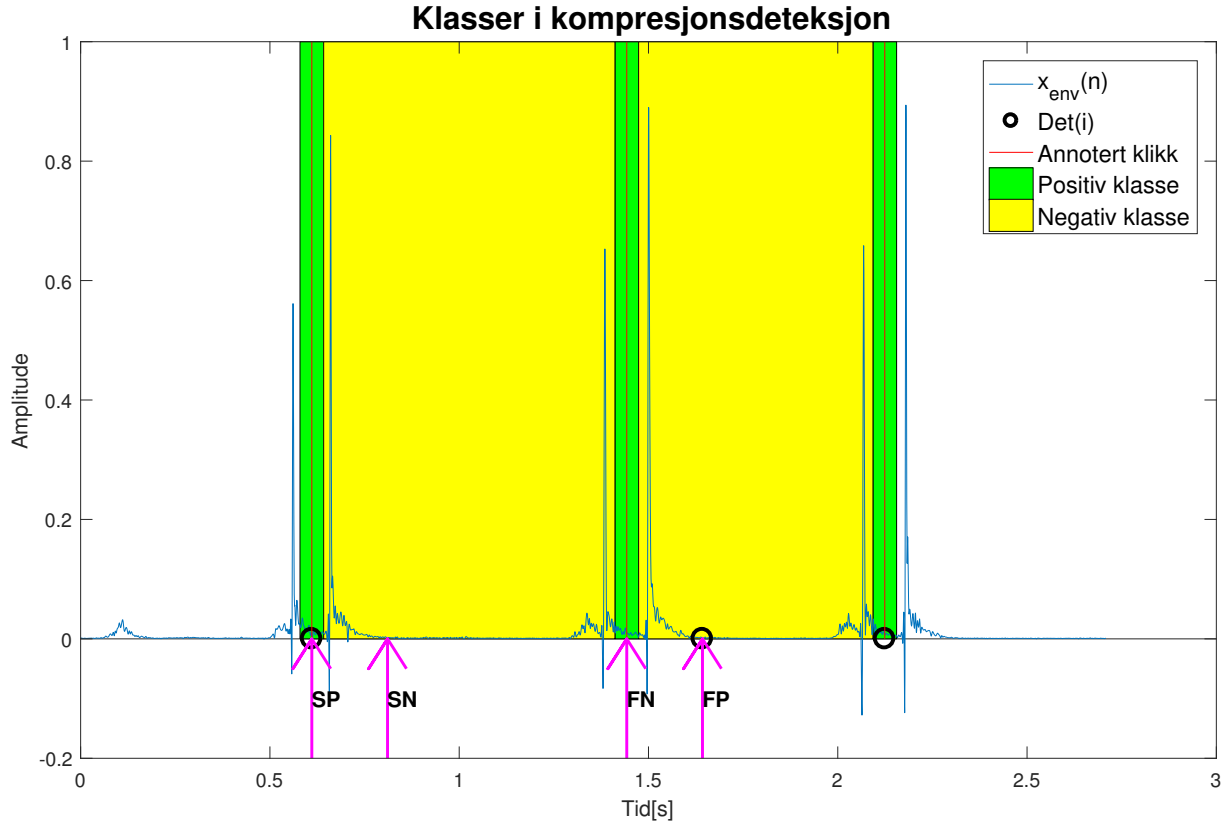
I denne oppgaven finnes envelopen ved å interpolere mellom toppunkt til signalet slik figur 2.5 viser.



Figur 2.5: Den røde grafen viser viser øvre envelope til det blå signal [20]

2.4 Ytelsesmål

Deteksjon av kompresjoner kan ses på som et to-klasse problem hvor vi har en positiv klasse og en negativ. Positiv klasse vil være et området hvor det er annotert kompresjon og negativ klasse vil være området hvor det ikke er annotert kompresjon. Det er her brukt et område O_p på 3 000 sampler rundt annotert kompresjon som positiv klasse, se grønt område i figur 2.6. Negativt område, O_n, varierer endel i størrelse. Negativt område ble derfor delt opp i områder av samme størrelse som O_n. Eks: dersom avstand mellom kompresjon 1 og kompresjon 2 er 22 000 sampler vil det være $\frac{22\,000}{3\,000} = 7.33 \approx 7$ negative områder O_n mellom kompresjon 1 og kompresjon 2. Dermed vil negativ klasse være vesentlig større enn positiv klasse.



Figur 2.6: Viser hvordan klassifiseringen er gjort.

For å kunne presentere resultatene på en god måte ble det brukt forvirringsmatrise (Confusion Matrix), eller feil-matrise. Dette er en mye brukt metode innenfor maskinlæring for å vise hvor godt en algoritme fungerer [21]. Forvirringsmatrise kan brukes så lenge man har resultater fra flere klasser. Tabell 2.1 viser et eksempel på en forvirringsmatrise. Her er

virkelige klasser radene i matrisen og predikterte klasser kolonnene. P står for positiv klasse og N for negativ klasse.

- **SP**: Sann Positiv betyr at det er detektert kompresjon i et positivt område hvor det er annotert kompresjon.
- **FP**: Falsk Positiv betyr at det er detektert kompresjon i et negativt område hvor det ikke er annotert kompresjon.
- **FN**: Falsk Negativ betyr at det ikke er detektert kompresjon i et positivt område hvor det er annotert kompresjon.
- **SN**: Sann Negativ betyr at det ikke er detektert kompresjon i et negativt område hvor det ikke er annotert kompresjon.

		Prediktert	
		P	N
Virkelig	P	SP	FP
	N	FN	SN

Tabell 2.1: Viser eksempel på en forvirringsmatrise (Confusion Matrix)

Ut av forvirringsmatrisen kan man hente flere mål på hvor god algoritmen er. Total ytelse blir målt ut i fra total nøyaktighet (tAcc), positiv prediktiv verdi (PPV), negativ prediktiv verdi (NPV), sensitivitet (Sen) og spesifitet (Spe):

- tAcc, total nøyaktighet: $\frac{SP+FP+FN+SN}{SP+SN}$. tAcc forteller hvor god totalnøyaktigheten er for alle klassene.
- PPV, Positiv Prediktiv Verdi: $\frac{SP}{SP+FP}$. PPV forteller hvor god nøyaktighet det er i positiv klasse. I dette tilfelle forteller det hvor stor andel av detekterte kompresjoner som var innenfor positivt område rundt en annotert kompresjon.
- NPV, Negativ Prediktiv Verdi: $\frac{SN}{SN+FN}$. NPV forteller hvor god nøyaktighet det er i negativ klasse. I dette tilfelle forteller det hvor stor andel av negativt område som ikke har detekterte kompresjoner.

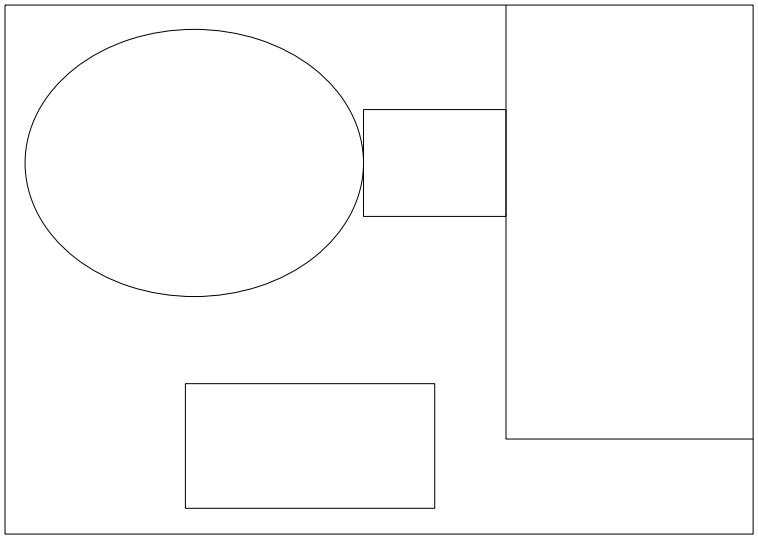
- Sen, Sensitivitet eller følsomhet: $\frac{SP}{SP+FN}$. Sen måler andelen positive som blir riktig klassifisert som positiv. I dette tilfellet forteller Sen hvor stor andel av positivt området hvor det er annotert kompresjoner der det blir detektert kompresjoner.
- Spe, Spesifisitet: $\frac{SN}{SN+FP}$. Spe måler andelen negative som blir riktig klassifisert som negativ. I dette tilfellet forteller Spe hvor stor del av negative områder hvor det ikke er detektert kompresjoner.

Kapittel 3

Datamateriell

Dette kapittelet beskriver datamateriellet som er brukt i denne masteroppgaven. Alt datamateriell er samlet inn underveis både alene og ved hjelp av familie og ansatte på Laerdal. Opptakene er gjort med forskjellige smarttelefoner som ligger plassert foran dukken og filmer opp i ansiktet på personen som utfører HLR trening. Det er kun lyden fra videoen som blir brukt for å detektere kompresjonene. Videoen ble kun brukt som støtte ved annotering av datasettene. For at plasseringen av telefonene skulle være tilnærmet lik i alle opptakene ble det designet en mal med omriss til dukke og telefon, se figur 3.1. Denne ble skrevet ut i A3 format og lagt under dukken i alle opptakene.

For å få tall på hvor god deteksjonen av klikkelydene var, ble hver kompresjon i Datasett Laerdal annotert. I Datasett 5 og Datasett 6 ble hver kompresjon annotert i tillegg til klikk ned og klikk opp for hver kompresjon.



Figur 3.1: Dette omrisset av dukke og telefon ble skrevet ut i A3 format og brukt som underlag i alle opptakene.

3.1 Datasett Laerdal

"Datasett Laerdal" er det største datasettet som har vært tilgjengelig under denne master-oppgaven. Datasettet består av 14 forskjellige opptak med totalt 9 telefoner og 9 personer som komprimerte. Protokollen for "datasett Laerdal" ligger vedlagt på side 76. Her ble det gjort en god del forskjellige opptak for å teste hvor lett det er å detektere klikkelyder som tilhører riktig dukke. Blant annet brukte man forskjellig avstand mellom dukker, forskjellig kompresjonsrate og forskjellig underlag. Dette materiellet er også brukt for å teste foreslått metode sammen med datasett 5 og 6.

3.2 Datasett 1

Datasett 1 er et lite datasett hvor målet har vært å teste dynamikken til mikrofonen. Protokollen for datasett 1 ligger vedlagt på side 87.

3.3 Datasett 2

Datasett 2 består av opptak for å teste Blind Source Separation. Protokollen for datasett 2 ligger vedlagt på side 90.

3.4 Datasett 3

Datasett 3 består av 3 tester for å teste lydforskjeller knyttet til forskjellig dukke, telefon og hvordan man komprimerer. Test 1 går ut på å teste forskjeller mellom hver dukke. Tre telefoner ble brukt og man får da testet forskjeller knyttet til telfoner også. Test 2 går ut på å komprimere på forskjellige måter, hardt og løst, på skrå og på forskjellige plasser på dukken. Hensikten er å finne ut om det blir vanskeligere å detektere klikkelyder dersom folk komprimerer slik. Test 3 går ut på å teste om en høy lyd påvirker amplituden på lydsignalet over tid. Protokollen for datasett 3 ligger vedlagt på side 91.

3.5 Datasett 4

Datasett 4 består av enkle oppsett for å teste for eksempel sensitivitet på mikrofonen, støy knyttet til underlag osv. Protokollen for datasett 4 ligger vedlagt på side 94.

3.6 Datasett 5

Datasett 5 er et stort datasettet med 5 personer og 8 opptak. Her utførte alle fem 30 kompresjoner i forskjellig rate fra 90-110 pr. min. Alle opptakene hadde forskjellig avstand i lengde og høyderetning. Protokollen for datasett 5 ligger vedlagt på side 95.

3.7 Datasett 6

Datasett 6 er tatt opp for å verifisere resultatene fra datasett 5 og for å kunne anbefale en avstand i lengderetning. Det ble gjort seks opptak med en fast høyde H på 1.4m og en lengde L som varierer fra 0.8m til 1.8m, se figur 8.5. Som i datsett 5 komprimerte alle fem i forskjellig rate fra 90-110 pr. min. Protokollen for datasett 6 ligger vedlagt på side 98.

Kapittel 4

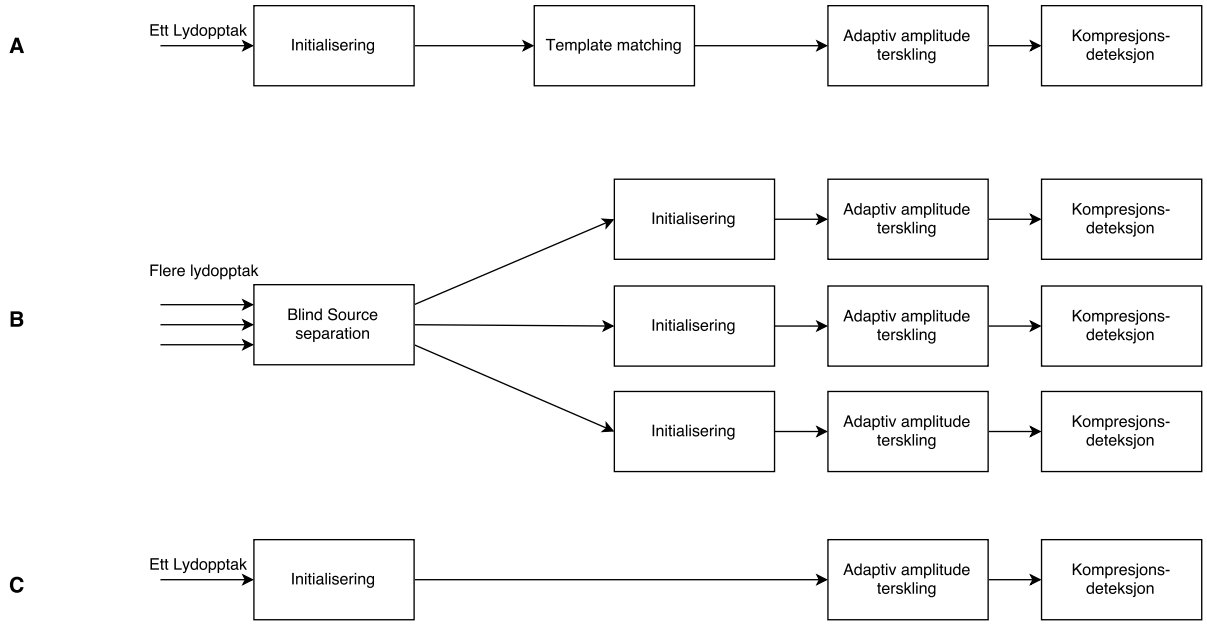
Metoder

Dette kapittelet tar for seg metodene som er brukt i eksperimentene og forklarer algoritmene som blir brukt. Figur 4.1 viser oversikt over tre forskjellige metoder å detektere kompresjoner på:

- A) Template matching
- B) Blind Source Separation
- C) Adaptiv amplitude terskling

Metode A og C gir tilbakemelding til bruker i sanntid mens metode B kun tillater tilbakemelding i ettertid eventuelt med en viss tidsforsinkelse. Hver av metodene har sine fordeler og ulemper basert på kompleksitet og evne til å detektere kompresjon. Før hver metode blir gjennomgått blir noen av algoritmene som blir brukt i flere metoder forklart i kapittel 4.1.

iPhone tar opp lyd i 44.1kHz mens Android telefoner tar opp lyd i 48kHz. For å få en fast samplerate på alle lydopptak blir lydopptak fra iPhone i denne oppgaven *resamplet* til 48kHz.



Figur 4.1: Oversikt over metodene som blir beskrevet i dette kapittelet

4.1 Algoritmer som brukes i flere metoder

Dette kapittelet tar for seg flere algoritmer som brukes flere ganger i metodene. Kapittelet består av følgende avsnitt:

- **Initialisering** utføres i alle metodene (A-C). De fem første kompresjonen en bruker utfører vil bli brukt i initialisering for å finne parametre som gjør deteksjonen av kompresjoner lettere.
- **Filtrering** utføres i alle metodene (A-C). Lydsignalet $x(n)$ filtreres med et båndpassfilter for å fjerne tale og annen uønsket støy.
- **Envelopen** til det båndpassfiltrerte signalet $x_{bp}(n)$ brukes under kompresjonsdeteksjon i stedet for $x_{bp}(n)$. Envelopen brukes i alle metodene (A-C).
- **Oppdatering av "terskelVerdi"** skjer adaptivt for å følge amplituden til lydsignalet, se figur 4.7. Den oppdateres likt i metode (B-C), mens i metode A oppdateres den på en litt annen måte, se kapittel 4.2.

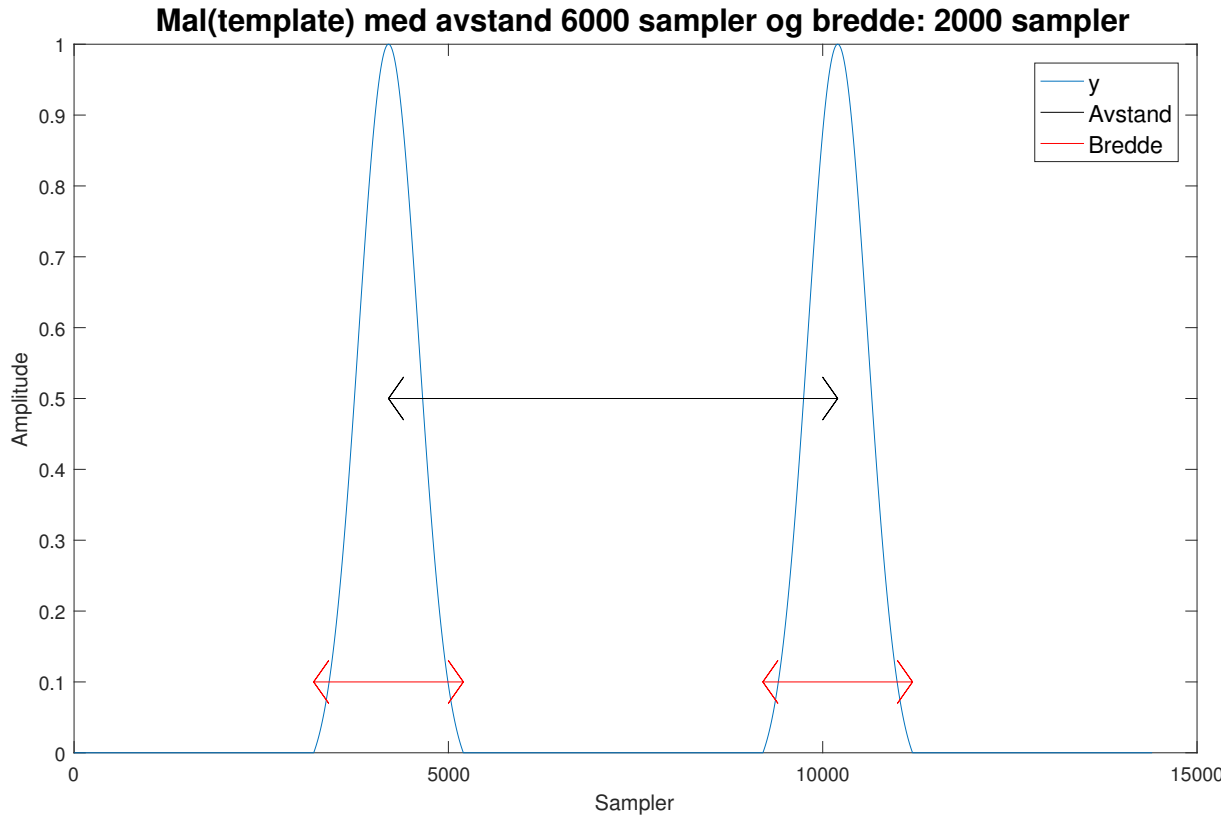
- **Kompresjonsdeteksjon**, selve deteksjonen av kompresjonene. Denne brukes i metode B-C og en litt annerledes versjon brukes i metode A.

4.1.1 Initialisering

Felles for alle metodene (A-C) er at det blir gjort en initialisering før selve kompresjonsdeteksjonen begynner. Bruker vil bli bedt om å gjøre fem kompresjoner helt i starten. Lydopptaket fra disse fem kompresjonene blir brukt for å finne parametre som brukes videre for å detektere klikkelyder. Følgende parametre finnes under initialisering:

- Kanal: Android telefoner tar opp lyd i to kanaler(stereo) mens iPhone tar opp lyd i kun én kanal. Den kanalen som det er lettest å detektere kompresjonene på blir valgt, se kapittel 5.3.
- "maksVerdi": gjennomsnittlig amplitude til toppunktene tilhørende 5 kompresjoner, gjennomsnitt av $Det_{amp}(1:5)$.
- "terskelVerdi": 0.3*"maksVerdi". I metode A, *template matching*, brukes 0.6*"maksVerdi". "terskelVerdi" brukes under kompresjonsdeteksjon, se kapittel 4.1.5.
- "toppunktMin" er gjennomsnitt av laveste toppunkt i hver tidligere godkjente kompresjon. "toppunktMin" brukes under kompresjonsdeteksjon, se kapittel 4.1.5.
- $Det_{amp}(i)$, amplitude på kompresjon $Det(i)$. Finnes ved å ta gjennomsnitt av amplituden til klikk ned og klikk opp av $x_{env}(n)$.
- Mal: Ved bruk av *template matching*, metode A, blir det laget en mal under initialiseringen. Denne består av to gaussfunksjoner med en bredde på 2000 sampler. Avstanden mellom gaussfunksjonene bestemmes av gjennomsnitt av avstand mellom klikk ned og klikk opp av de fem kompresjonene. Et eksempel på malen er vist i figur 4.2. Røde piler markerer bredde på gaussfunksjonene og svart pil markerer avstanden mellom gaussfunksjonene.

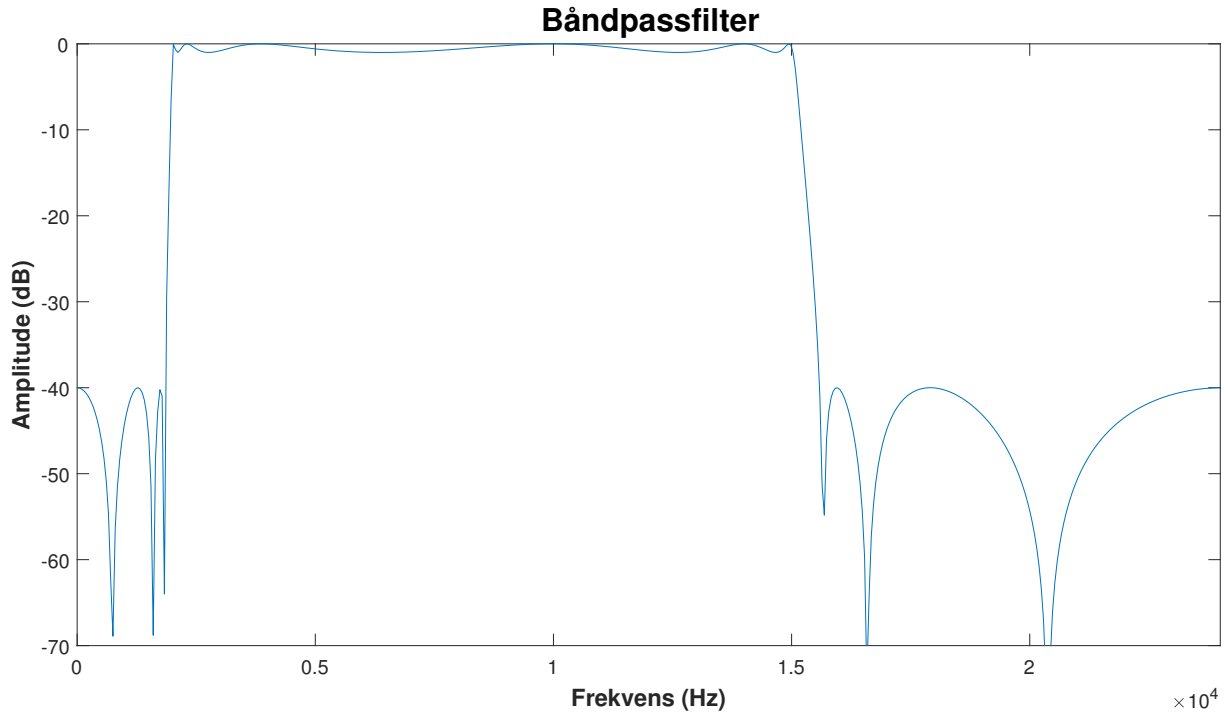
Initialisering utføres kun èn gang helt i starten.



Figur 4.2: Mal(template) generert under initialisering til bruk i *template matching*

4.1.2 Filtrering

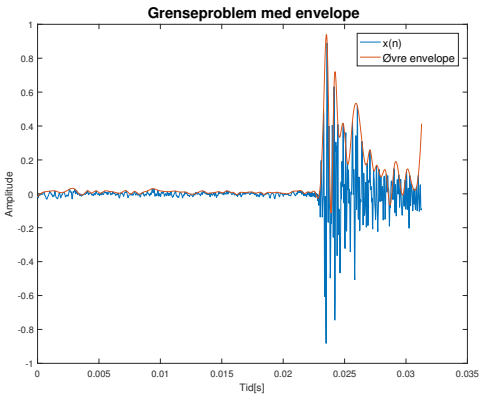
I alle metodene blir signalet filtrert med et bredt båndpassfilter. Dette gjøres for å filtrere bort uønsket støy. Eksempler på uønsket støy kan være tale eller støy fra dukke eller underlag. Båndpassfilteret er bygget opp slik at frekvenser mellom 2 000Hz og 15 000Hz slippes gjennom slik figur 4.3 viser. Frekvenser lavere enn 1 000Hz eller høyere enn 16 000Hz blir dempet med 40dB. Begrunnelser for valg av båndpassfilter er gitt i kapittel 5.5.



Figur 4.3: Frekvensrespons til båndpassfilteret

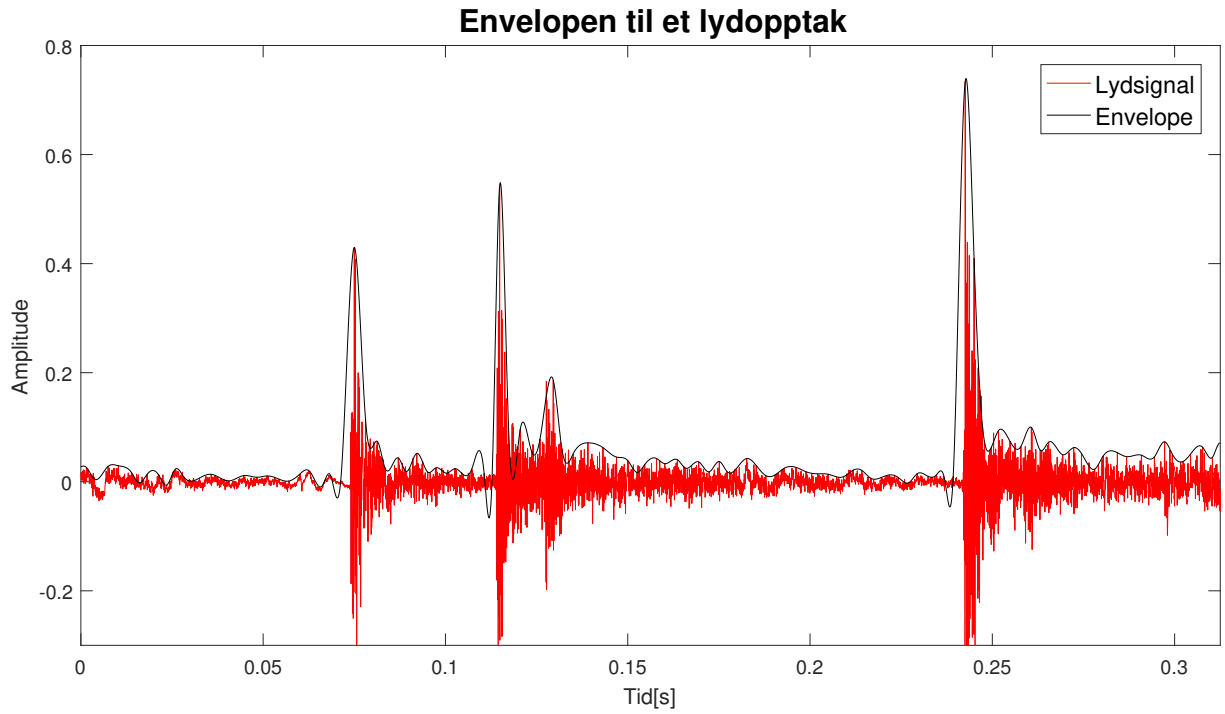
4.1.3 Envelope

For å lettere kunne utfør kompresjonsdeteksjon blir øvre envelope til lydsignalet brukt under kompresjonsdeteksjon istedenfor lydsignalet selv. Figur 4.5 viser øvre envelope som følger toppunktene til lydsignalet. Envelopen finnes ved å interpolere mellom lokale toppunkt til signalet, se kapittel 2.3. Her er det valgt å interpolere mellom lokale toppunkt som er separert med minst 100 sampler. Dette for å få én topp for hvert klikk og ikke flere. Interpolering kan føre til problemer ved endepunktene. Figur 4.4 viser $x(n)$ til ett klikk sammen med øvre envelope. Helt i grenseområdet i slutten stiger envelopen opp til 0.4 som følge av interpoleringen, selv om $x(n)$ holder seg rundt 0. For å unngå dette problemet blir $x(n)$



Figur 4.4: Grenseproblem med envelope

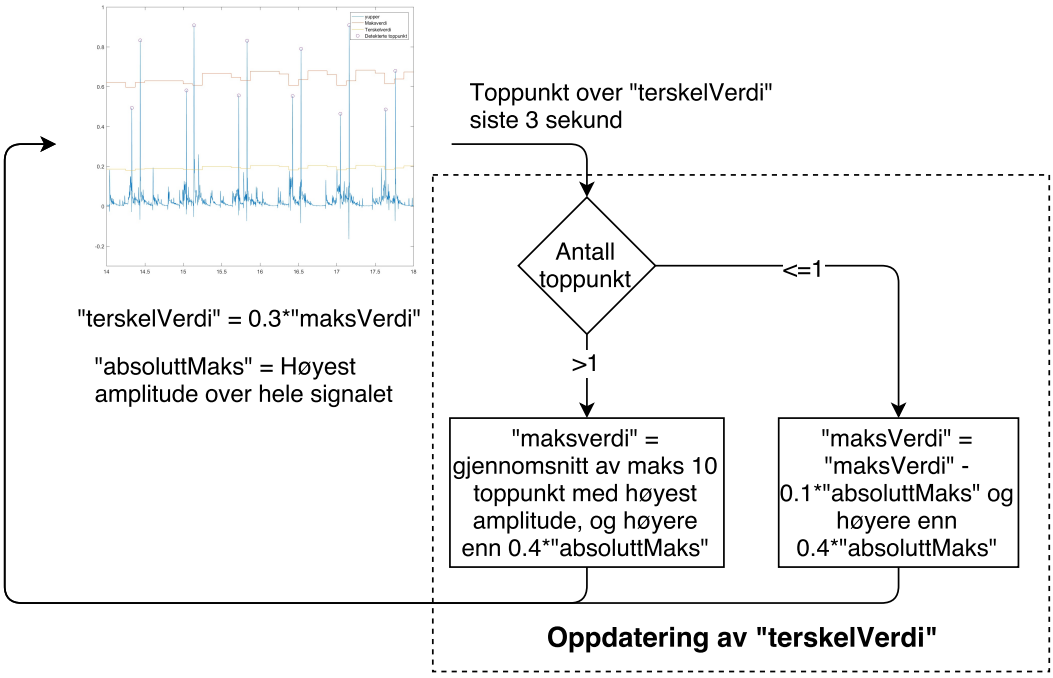
forsinket med 6 000 sampler. Dermed er det alltid 6 000 sampler frem i tid og bak i tid som kan brukes når under interpolering.



Figur 4.5: Envelopen til et lydsignal

4.1.4 Oppdatering av "terskelVerdi"

Terskling brukes for å kunne vite hvor høy amplitude hver klikkelyd må ha for å kunne tilhøre nærmeste dukke. Alle toppunkt til $x_{env}(n)$ over terskelverdien detekteres og brukes i kompresjonsdeteksjon, se kapittel 4.1.5. Resultater fra kapittel 5.1 viser at amplituden til klikkelyder fra en dukke kan være forskjellig fra amplituden til klikkelyder fra en annen dukke. I tillegg er det stor variasjon i amplituden til klikkelyder fra telefon til telefon. Derfor ble algoritmen for å finne terskelverdi gjort adaptiv slik figur 4.12 viser.



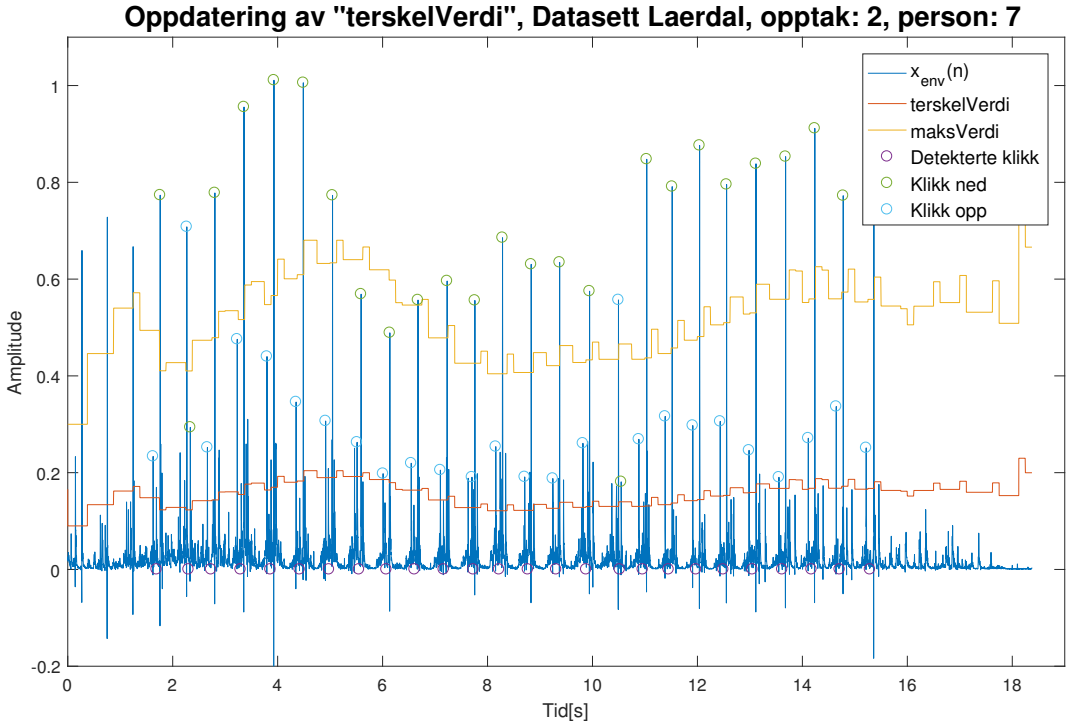
Figur 4.6: Figuren viser algoritmen for adaptiv amplitude terskling

Inngangen til oppdatering av “terskelVerdi”-algoritmen er detekterte toppunkt fra tre siste sekund og utgangen er maksverdi. “terskelVerdi” er gitt som 30% av “maksVerdi”.

- Dersom antall toppunkt inn i algoritmen er mer enn èn vil “maksVerdi” bli satt til gjennomsnittet av toppunktene. Dersom antallet toppunkt er mer enn ti blir kun de ti med høyest amplitude brukt. Tallet ti er valgt fordi kompresjoner 100 ganger i minuttet, som er anbefalt, gir ti toppunkt på tre sekund. Uansett kan ikke maksverdien være mindre enn 40% av den høyeste amplituden som har vært over hele lydopptaket, “absoluttMaks”. Dette for å hindre at “terskelVerdi” går langt ned dersom personen tar pause i komprimering og klikkelyder fra nabodukkene blir detektert.

- Dersom antall toppunkt er mindre enn eller lik én, settes “maksVerdi” lik “maksVerdi”-0.1*“absoluttMaks”, men ikke mindre enn 40% av “absoluttMaks”.

Adaptiv amplitude terskling brukes i alle metodene (A-C). Metode A bruker maks fem toppunkt siste tre sekund istedet for ti, da $x_{conv}(n)$ gir én topp for hver kompresjon i stedet for to. Figur 4.7 viser hvordan terskelverdien følger amplituden til klikkelydene. Vanligvis varierer ikke amplituden på klikkelydene i nærheten av så mye som figur 4.7 viser.



Figur 4.7: Figuren viser hvordan "maksVerdi" og "terskelVerdi" følger amplituden på lydopptaket

4.1.5 Kompresjonsdeteksjon

Kompresjonsdeteksjon er den viktigste delen av metodene (A-C), se figur 4.1. Det er her det avgjøres om klikkelydene er en kompresjon eller ikke. Algoritmen for deteksjon av kompresjon er vist i figur 4.8. I metode B og C er algoritmen lik, mens i metode A er det kun ett toppunkt som detekteres i stedet for 2, se kapittel 4.2. Hver boks i algoritmen i figur 4.8 er nummerert og består av:

1. Algoritmen for å detektere kompresjoner starter med et vindu med $x_{env}(n)$ på 6 000 sampler (125ms) som forskyver seg med 6 000 sampler mellom hver iterasjon. På figur 4.8 er vinduet markert med grønt.

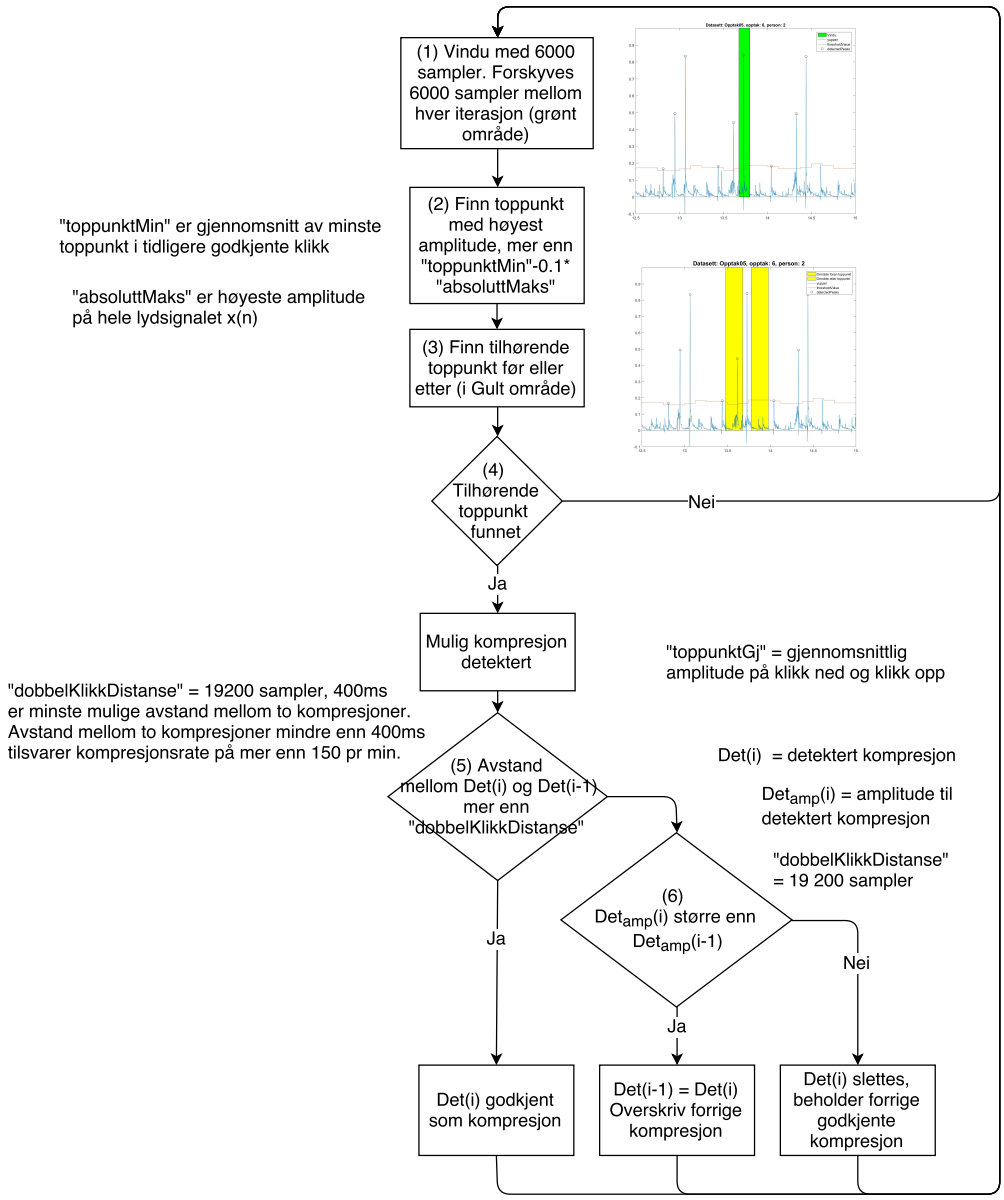
2. Høyeste toppunkt inni vinduet detekteres og det antas å tilhøre riktig dukke dersom amplituden er høyere enn "toppunktMin" -0.1*"absoluttMaks". "toppunktMin" er gjennomsnitt av laveste toppunkt i hvert tidligere godkjente klikk.

3. Det blir lett etter toppunkt med amplitude over "terskelVerdi" i et område foran og et område bak høyeste toppunkt fra forrige punkt. Dette området er markert i gult i figur 4.8 og områdets størrelse er gitt av "enkelKlikkDistanseMin" og "enkelKlikkDistanseMaks". Verdien på disse kan finnes i kapittel 5.4.

4. Dersom det blir funnet et toppunkt i det gule området, er det muligens funnet en kompresjon. Om den blir godkjent eller ei avgjøres i de to neste punkt. Dersom det ikke blir funnet et toppunkt i det gule området avsluttes algoritmen.

5. Avstanden mellom hver kompresjon må være mer enn 0.4 sekund. Dette tilsvarer en kompresjonsrate på mindre enn 150 pr minutt. Dersom avstanden mellom forrige kompresjon og denne er mer enn 0.4 sekund vil det bli registrert som godkjent kompresjon.

6. Dersom avstanden mellom forrige kompresjon og gjeldende er mindre enn 0.4 sekund sammenlignes gjennomsnitt av amplitude på klikk ned og klikk opp for hver av de to siste kompresjonene. Den kompresjonene med høyest gjennomsnittlig amplitude vil bli valgt som godkjent kompresjon.

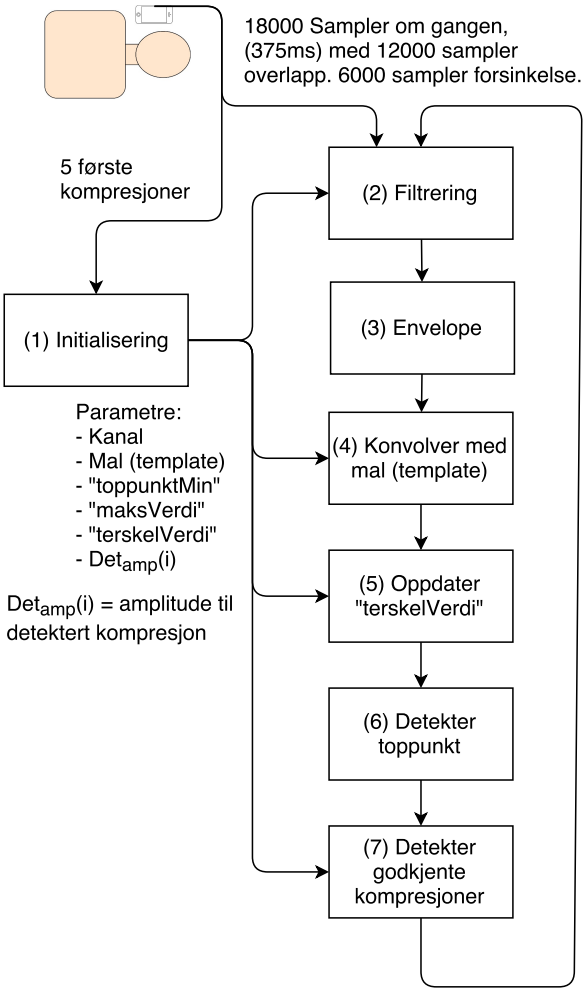


Figur 4.8: Algoritmen for kompresjonsdeteksjon

4.2 Metode A: Template Matching

Metode A fra figur 4.1 beskriver hvordan *Template Matching* ble utført i praksis. Et mer detaljert skjema over algoritmen er vist i figur 4.9.

Før selve kompresjonsdeteksjonen starter må det komprimeres 5 ganger. Dette brukes så under initialisering (1) som forklart i kapittel 4.1.1. Man får da en del parametre som brukes videre i algoritmen for å detektere kompresjoner, se figur 4.9. Algoritmen får et vindu på 18 000 sampler om gangen med 12 000 sampler overlapp, $x(n)$. Siden største avstand mellom klikk ned og klikk opp, "enkelKlikkDistanseMaks", er på 12 000 sampler er man da helt sikker på at alle kompresjonene vil være helt innenfor ett vindu. $x(n)$ blir filtrert (2) med båndpassfilter, se kapittel 4.1.2, $x_{bp}(n)$ blir brukt for å finne øvre envelope (3) $x_{env}(n)$. $x_{env}(n)$ blir konvolvert (4) med malen som er funnet under initialisering. "terskelVerdi" (5) blir så oppdatert basert på tidligere toppunkt. Deteksjonen av toppunktene (6) gjøres på en litt annerledes måte enn de andre metodene. Her får man én høy topp dersom malen stemmer overens med $x_{env}(n)$ figur 2.4 viser. Det blir til slutt (7) avgjort om toppunktet skal godkjennes som en kompresjon $Det(i)$. Algoritmen til de tre nederste blokkene i figur 4.9 er vist i detalj i figur 4.10. Her representerer en stiplet linje en blokk i figur 4.9.



Figur 4.9: Flytskjema over Template Matching

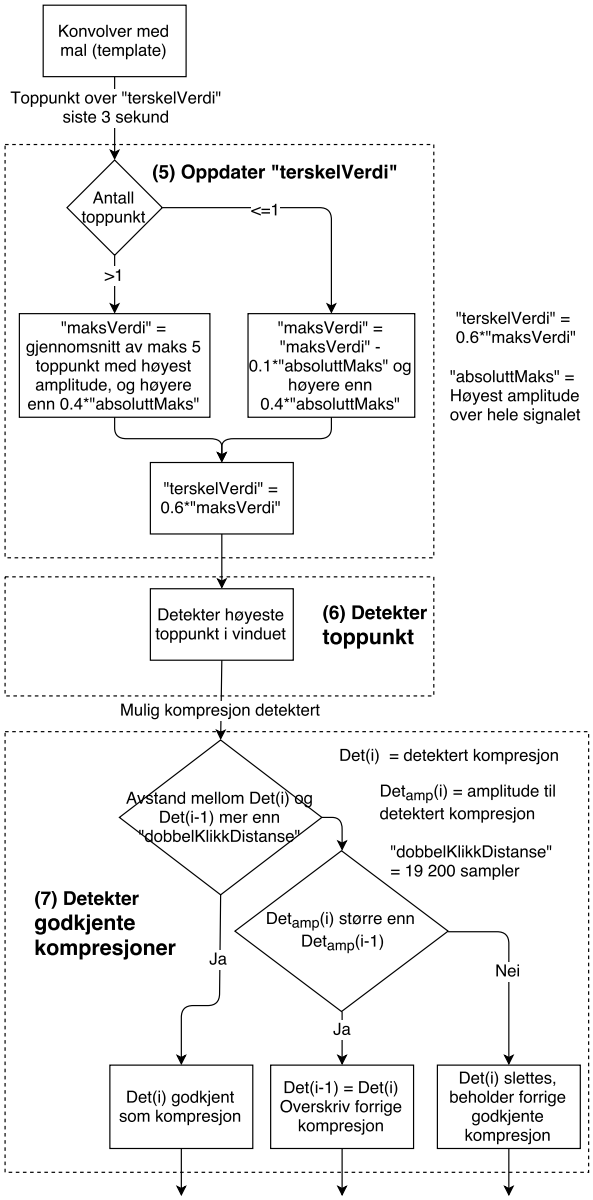
Første blokk i figur 4.10, oppdatering av "terskelVerdi", gjøres basert på toppunkt med amplitude over "terskelVerdi" siste tre sekund. "maksVerdi" settes lik gjennomsnitt av opptil

fem toppunkt over “terskelVerdi” siste tre sekund. Dersom det er ett eller færre toppunkt med amplitude over “terskelVerdi” siste tre sekund blir “maksVerdi” satt lik “maksVerdi” - 0.1*“absoluttMaks”. “maksVerdi” vil hele tiden være høyere enn 0.4*“absoluttMaks”. “terskelVerdi” settes så lik 0.6*“maksVerdi”.

Blokk to i figur 4.10, detekter toppunkt, gjøres ved å finne høyeste amplituden innenfor vinduet. Vinduet er på 18 000 sampler med 12 000 sampler overlapp. Dermed vil det kun være én topp innenfor vinduet som kan være en kompresjon.

Siste blokk i figur 4.10, detekter godkjente kompresjoner, gjøres på tilsvarende måte som kompresjonsdeteksjon i figur 4.8. Avstanden mellom hver kompresjon må være mer enn 0.4 sekund. Dette tilsvarer en kompresjonsrate på mindre enn 150 pr minutt. Dersom avstanden mellom forrige kompresjon og denne er mer enn 0.4 sekund vil det bli registrert som godkjent kompresjon. Dersom avstanden mellom forrige kompresjon og denne er mindre enn 0.4 sekund sammenlignes gjennomsnitt av amplitude på klikk ned og klikk opp for hver av de to siste kompresjonene. Den kompresjonene med høyest gjennomsnittlig amplitude vil bli valgt som godkjent kompresjon.

Algoritmen kjøres så på nytt med nytt vindu forskjøvet 6 000 sampler.

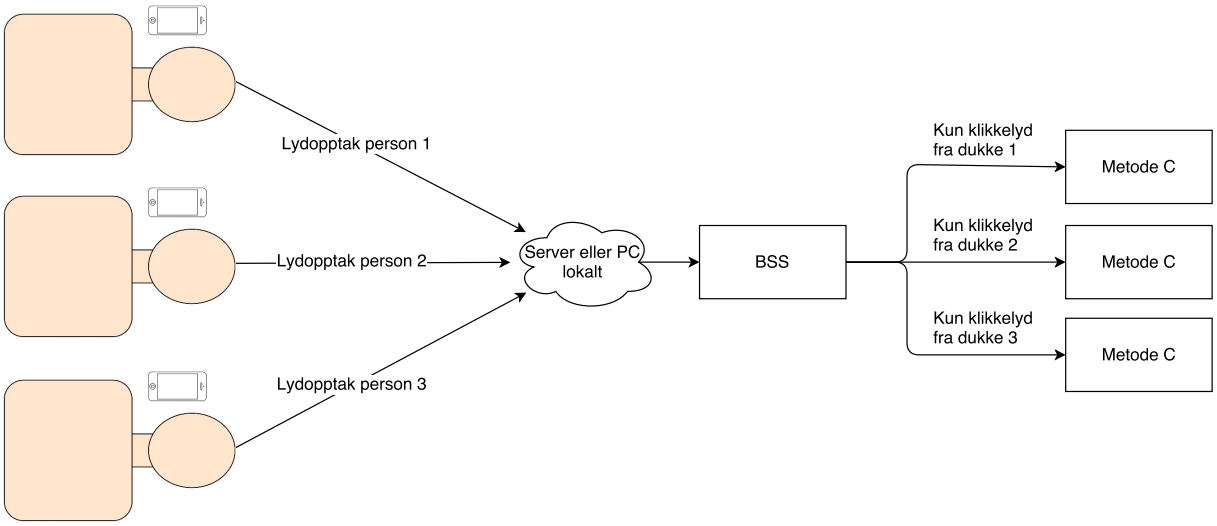


Figur 4.10: Viser algoritmen til de tre nederste boksene i figur 4.9

4.3 Metode B: Blind Source Separation

Blind Source Separation (BSS) er metode B i figur 4.1. Dette er den mest krevende av alle metoder, både i form av utstyr som kreves og i form av kompleksitet. Denne metoden er vanskelig å utføre i sanntid. Fra ligning 2.5 i kapittel 2.1 må man finne miksematrisen **W** for å kunne finne tilbake til **U**, et estimat av kildesignalene. Dersom miksematrisen **W** først er funnet kan denne i teorien brukes videre i hele signalet. Dette forutsetter at romresponsen er konstant. Da romresponsen vil variere med tiden, vil også **W** variere med tiden og **W** må oppdateres underveis.

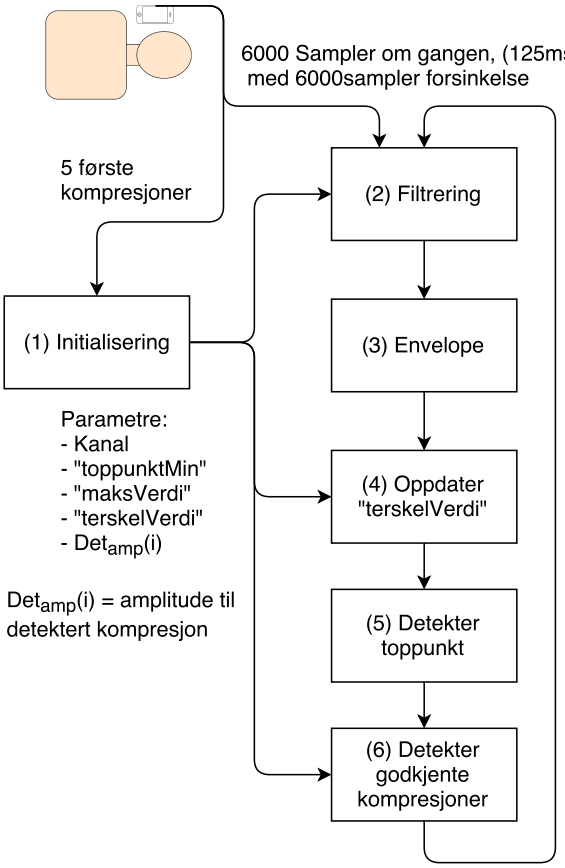
Metoden for å utføre BSS er vist i figur 4.11. Hele, eller deler av lydopptaket vil bli sendt til en server eller en PC som utfører BSS. Da vil man kunne finne kildelydene i de miksede lydopptakene som mikrofonen tar opp, se kapittel 2.1. Algoritmen som brukes til å utføre BSS er FastICA [9] utviklet ved universitetet i Helsinki. Kildelydene i denne sammenheng er klikkelyder tilhørende èn dukke. Når kildelydene er funnet er det lett å detektere kompresjoner ved hjelp av metode C, adaptiv amplitude terskling, som er beskrevet i kapittel 4.4.



Figur 4.11: Figuren viser hvordan BSS kan utføres

4.4 Metode C: Adaptiv amplitude terskling

Adaptiv amplitude terskling er metode C vist i figur 4.1. Denne metoden er den minst kompliserte av alle metodene. Algoritmen til adaptiv amplitude terskling er vist i figur 4.12. De fem første kompresjonene blir brukt under initialisering (1) som forklart i kapittel 4.1.1. Dette gir følgende parametre: Kanal, "toppunktMin", "maksVerdi", "terskelVerdi" og $Det_{amp}(i)$. Når initialiseringen er ferdig mottar adaptiv amplitude terskling-algoritmen et vindu, $x(n)$ på 6 000 sampler om gangen. Det er lagt inn 6 000 sampler forsinkelse for å kunne finne envelopen. Algoritmen består av følgende steg, se figur 4.12:



Figur 4.12: Viser algoritmen Adaptiv amplitude terskling

- Filtrering: (2) $x(n)$ blir filtrert av et båndpassfilter som forklart i kapittel 4.1.2.
- Envelope: (3) Envlopen til det båndpass-filtrerte signalet $x_{bp}(n)$ blir funnet som forklart i kapittel 4.1.3.
- Oppdater "terskelVerdi": (4) "terskelVerdi" oppdateres adaptivt basert på tidligere toppunkt som har høyere amplitude enn "terskelverdi", se kapittel 4.1.4.
- Detekter toppunkt: (5) Toppunkt til $x_{env}(n)$ som har høyere amplitude enn "terskelVerdi" blir detektert og er mulige kandidater til en kompresjon.
- Detekter godkjente kompresjoner: (6) Her gjøres selve kompresjonsdeteksjonen. Denne er forklart i kapittel 4.1.5.

Kapittel 5

Eksperiment og resultat

Dette kapittelet gjennomgår eksperimentene som er utført i oppgaven. Alle eksperimentene er gjengitt i listen under.

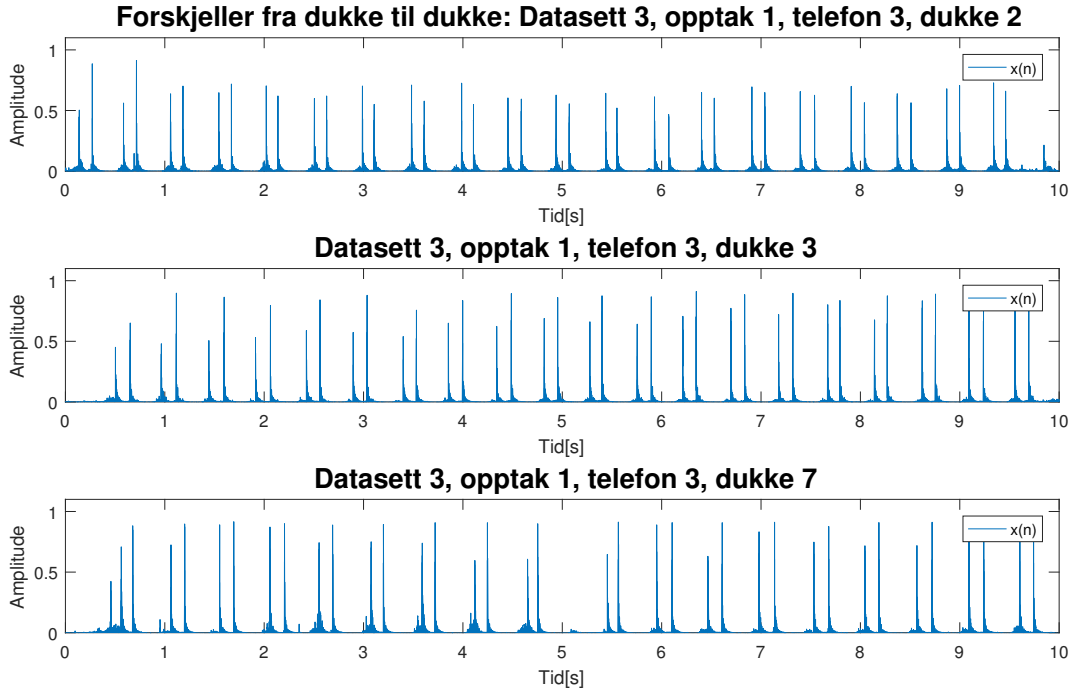
- Eksperiment 1: Test av dukkelyder
- Eksperiment 2: Test av telefoner
- Eksperiment 3: Kanaldeteksjon
- Eksperiment 4: Valg av forskjellige parametre
- Eksperiment 5: Valg av båndpassfilter
- Eksperiment 6: Valg av avstand mellom gaussfunksjoner i mal
- Eksperiment 7: Valg av bredde på gaussfunksjoner i mal
- Eksperiment 8: Blind source separation
- Eksperiment 9: Template Matching
- Eksperiment 10: Adaptiv amplitude terskling

- Eksperiment 11: Påvirkning fra nabodukker

Alle eksperimentene er implementert i Matlab. Det er i tillegg brukt FastICA toolbox [9] for å utføre *blind source separation*.

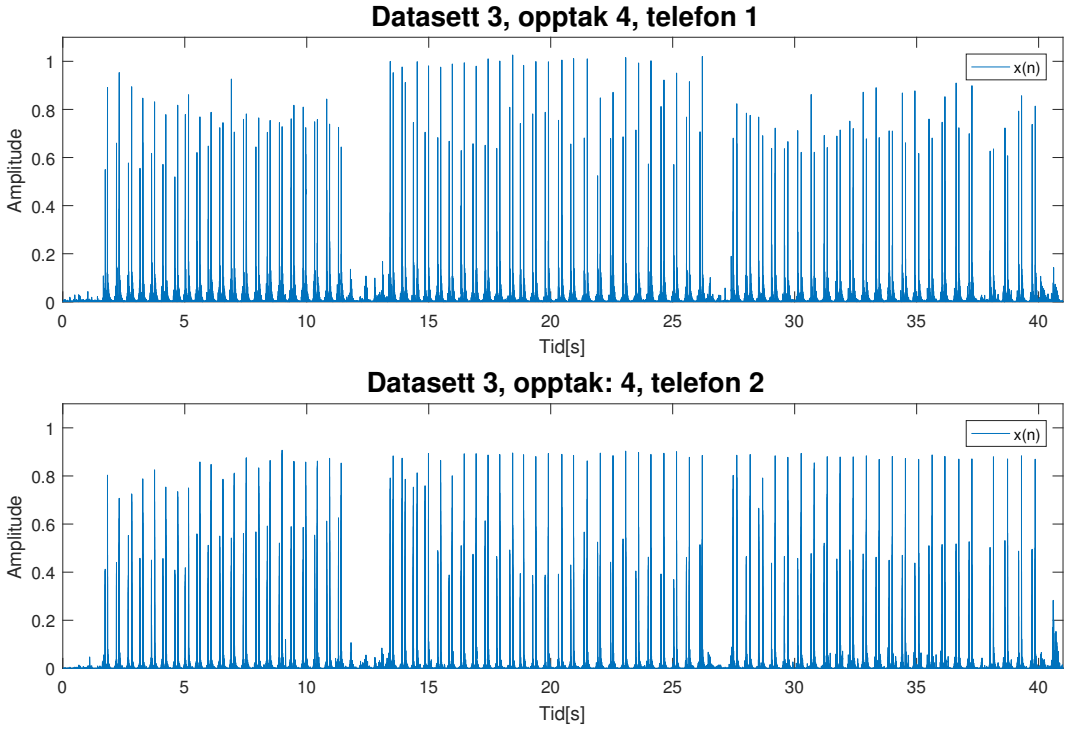
5.1 Eksperiment 1: Test av dukkelyder

Alle MiniAnne-dukkene inneholder samme "klikker" som lager klikkelyder når man komprimerer, se figur 1.1b. Likevel er lyden fra de individuelle dukkene litt forskjellig. Det er ikke lett å høre forskjell på lyden, men dersom man ser på plot av lydopptaket fra forskjellige dukker, kan man se at amplituden varierer. Lyden kan variere fra klikk til klikk og fra dukke til dukke. Figur 5.1 viser plot av lydopptak fra kompresjoner på dukke 2, 3 og 7. For å generere denne figuren er det brukt opptak 1 og telefon 3 fra datasett 3, se side 91. De tre dukkene som gav mest forskjellig amplitude er de som er vist i figur 5.1.



Figur 5.1: Forskjell i lyd mellom dukke 2, 3 og 7. Alle tre lydopptakene er gjort i samme opptak og med samme telefon

En annen måte for å teste dukkelyder gikk ut på å prøve å fremprovosere lyder som kan gjøre det vanskelig å detektere kompresjoner. For å få til dette ble det komprimert på skrå og litt utenfor midten. Figur 5.2 viser opptak 4 i datasett 3, se side 91.



Figur 5.2: I disse opptakene ble kompresjoner utført på forskjellig vis for å fremprovosere variasjon i klikkelyd.

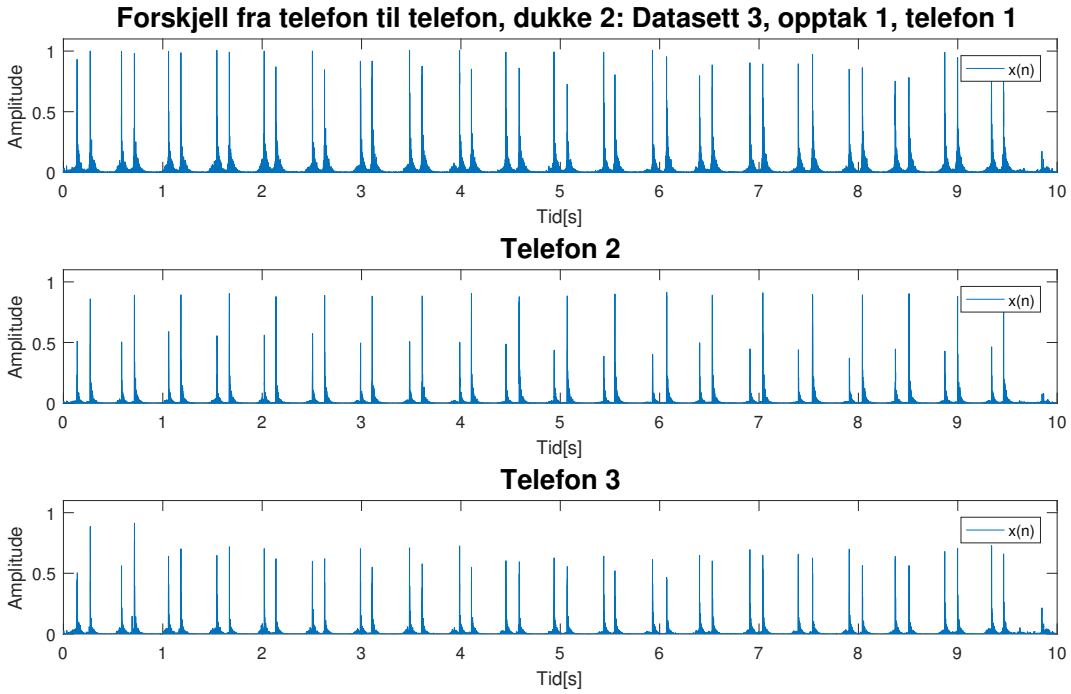
Resultat

Figur 5.1 viser at amplituden på lyden til dukke 2 er en del lavere enn amplituden på lyden til dukke 3 og 7. Dukke 3 gir lavere amplitude på klikk ned i begynnelsen men endres gradvis. Dukke 7 gir lik amplitude på klikk ned og klikk opp noen ganger mens andre ganger har klikk ned lavere amplitude. Figur 5.2 viser at det er litt forskjellig amplitude på klikkelydene avhengig av hvordan man komprimerer.

Totalt sett viser resultatene at amplituden på klikkelyden kan variere med forskjellige dukker. Det er også vist at måten man trykker på betyr mindre enn hvilken dukke man benytter, men det er noen forskjeller fra klikk til klikk også på samme dukke.

5.2 Eksperiment 2: Test av telefoner

Det finnes stadig flere forskjellige typer smarttelefoner. Disse har gjerne forskjellig mikrofon og elektronikk i tillegg til forskjellig utforming. Et lydopptak tatt med en telefon vil gjerne se forskjellig ut i forhold til et lydopptak tatt med en annen telefon. Figur 5.3 viser deler av opptak 1, i datasett 3, som består av 20 kompresjoner på dukke 2, se side 91. Dette opptaket ble gjort med tre telefoner samtidig. I figur 5.3 ser man hvor forskjellig amplituden på klikkelydene kan være på forskjellige telefoner.



Figur 5.3: Kompresjoner med dukke 2 tatt opp med tre telefoner samtidig.

Resultat

Lydopptak fra telefon 1 i figur 5.3 har høy amplitude og omtrent like høy amplitude på klikk ned som på klikk opp. Lydopptak fra telefon 2 har lav amplitude på klikk ned og høy amplitude på klikk opp. Lydopptak fra telefon 3 har lavere amplitude enn de to andre. Dette viser at hver telefon kan gi forskjellig amplitude på klikkelydene.

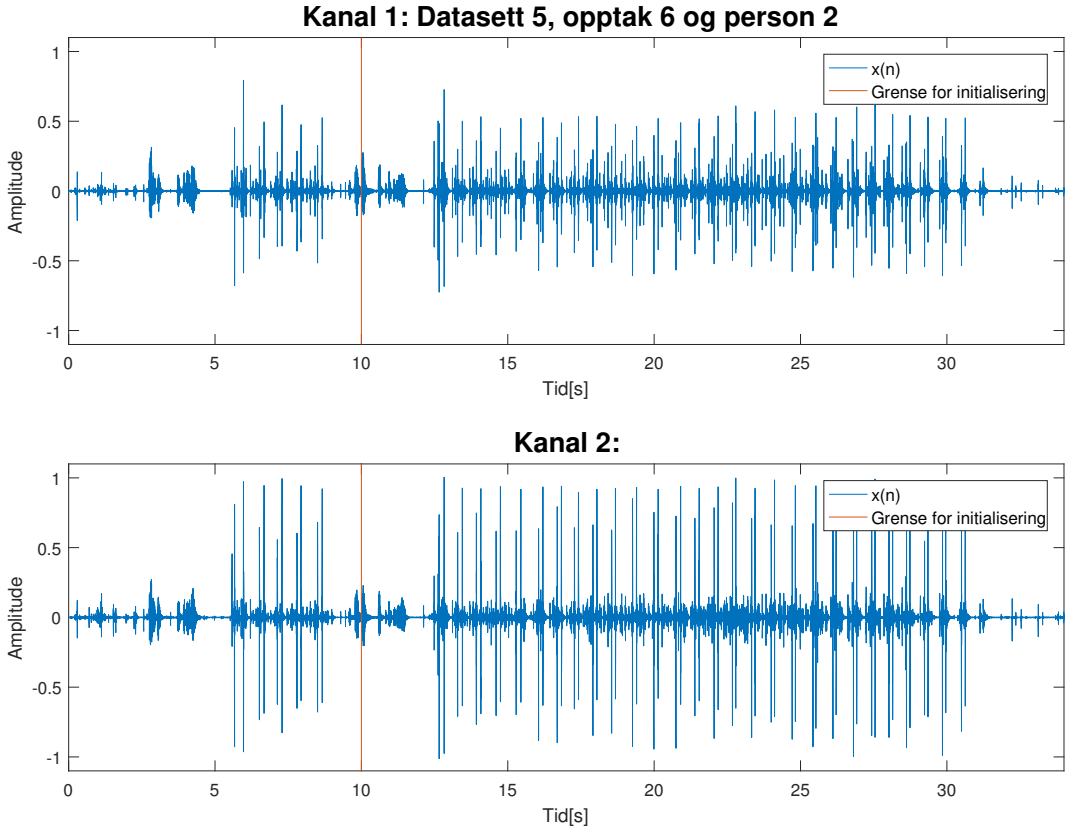
5.3 Eksperiment 3: Kanaldeteksjon

Til dette eksperimentet ble det brukt datasett 5, datasett 6 og datasett Laerdal. Disse datasettene er annotert med hvilken kanal som det er lettest å detektere kompresjoner fra. Den kanalen som er annotert som lettest å detektere kompresjoner fra er omtalt som riktig kanal.

Alle iOS telefoner tar opp lyd i kun én kanal (mono), mens Android-telefoner tar opp lyd i to kanaler (stereo). Noen android-telefoner tar opp lyd både ved topp og bunn av telefonen for å kunne redusere støy i opptaket. Forskjellen mellom de to kanalene kan være veldig stor, som vist med et eksempel i figur 5.4. I alle eksperimenter videre i denne oppgaven benyttes kun én lydkanal. For å kunne detektere klikk best mulig er det viktig at den beste kanalen blir valgt. Deteksjon av kompresjoner er lettest når klikk ned og klikk opp har såpass høy amplitude at de skiller seg fra klikk fra nabodukker. Fra nederste plottet i figur 5.4, kanal 2, kan man se at amplituden til klikk ned og klikk opp er vesentlig høyere enn amplituden på klikkelyder fra nabodukken. I øverste lydopptak, kanal 1, skiller klikk opp seg ut med høy amplitude mens klikk ned har nesten like lav amplitude som klikk fra nabodukkene. Dermed vil kompresjonsdeteksjon fungere best når kanal 2 blir valgt i dette tilfelle.

Kanaldeteksjon gjøres under initialisering etter at bruker har komprimert 5 ganger, se kapittel 4.1.1. Dermed er det fem kompresjoner tilgjengelig for å finne beste kanal. I datasett 5 og 6 ble det komprimert fem ganger helt i starten, se ti første sekund i figur 5.4. Denne delen av opptaket ble brukt til initialisering inkludert kanaldeteksjon. I datasett Laerdal ble det ikke komprimert fem ganger helt i starten. Derfor måtte hele lydopptaket benyttes under initialisering for å finne best mulig kanal.

I noen tilfeller har lydopptaket én høy topp som har en del høyere amplitude enn resten, slik som øverste lydopptak i figur 5.4 viser (rundt 6 sekund). Derfor brukes gjennomsnittlig amplitude av 5 høyest toppunkt som et mål på hvilken kanal som skal velges.



Figur 5.4: Eksempel på hvor forskjellig kanal 1 og kanal 2 kan være på android-telefoner

Resultat

Resultatene fra kanaldeteksjon er vist i tabell 5.1. Resultater viser at riktig kanal ble funnet i 97.5% av opptakene i datasett 5 og 100% av opptakene i datasett 6. Datasett Laerdal gir noe dårligere resultat hvor det kun er detektert riktig kanal i 82.0% av opptakene. Tabell 5.2 viser resultat av kanaldeteksjon i datasett Laerdal. Her ser man at det kun er person 5 og 8 som får detektert feil kanaler.

Resultater:	Datasett 5	Datasett 6	Datasett Laerdal
Antall opptak:	40	30	70
Antall riktig detekterte kanaler:	39	30	58
Prosent riktig detekterte kanaler:	97.5%	100%	82.86%

Tabell 5.1: Tabellen viser hvor mange opptak i hvert datasett hvor riktig kanal ble detektert.

Opptak/Person nr.	2	4	5	7	8
Opptak 1	1	1	0	1	0
Opptak 2	1	1	0	1	1
Opptak 3	1	1	0	1	1
Opptak 4	1	1	1	1	0
Opptak 5	1	1	0	1	0
Opptak 6	1	1	1	1	1
Opptak 7	1	1	0	1	1
Opptak 8	1	1	1	1	1
Opptak 9	1	1	1	1	1
Opptak 10	1	1	1	1	1
Opptak 11	1	1	0	1	1
Opptak 12	1	1	0	1	1
Opptak 13	1	1	1	1	0
Opptak 14	1	1	1	1	0

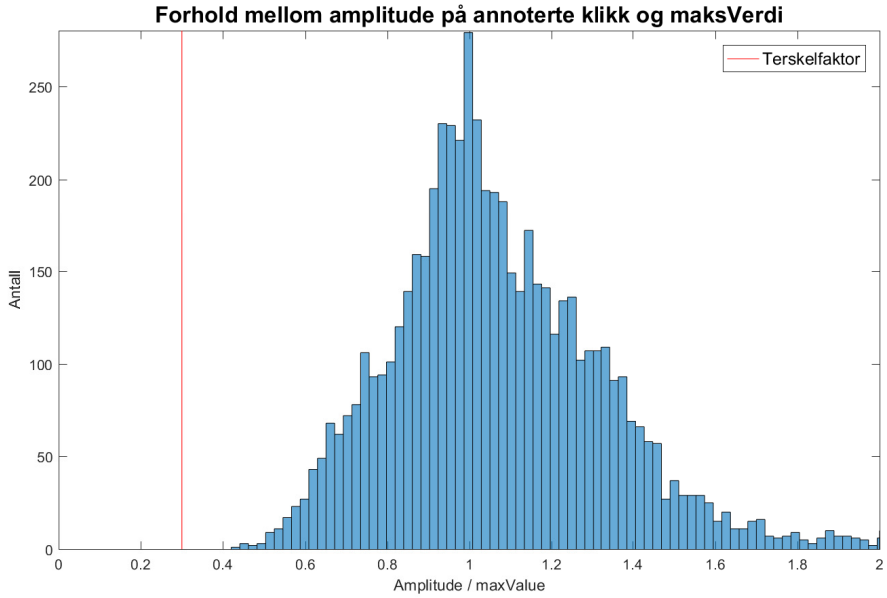
Tabell 5.2: Resultat av kanaldeteksjon i datasett Laerdal. 1 betyr at riktig kanal ble detektert og 0 betyr at feil kanal ble detektert. Person 1, 3 og 6 brukte iPhone som kun har én kanal og er derfor ikke tatt med i resultatene her.

5.4 Eksperiment 4: Valg av forskjellige parametre

Av parametrene som er nevnt i kapittel 1.4 er noen av disse globale konstanter. Dette delkapittelet vil forklare eksperimentene som er gjort for å finne verdien til disse parametrene.

"terskelFaktor"

"terskelFaktor" er faktoren som multipliseres med "maksVerdi" for å finne "terskelVerdi". For å finne best verdi på "terskelFaktor" ble alle annoterte klikk fra datasett 5 og 6 divitert med tilhørende "maksVerdi". Et histogram over disse verdiene er vist i figur 5.5. Med en "terskelFaktor" på 0.3 kan man være helt sikker på at alle klikk fra riktig dukke vil bli detektert. Dermed vil "terskelVerdi" være lik 0.3*"maksVerdi".



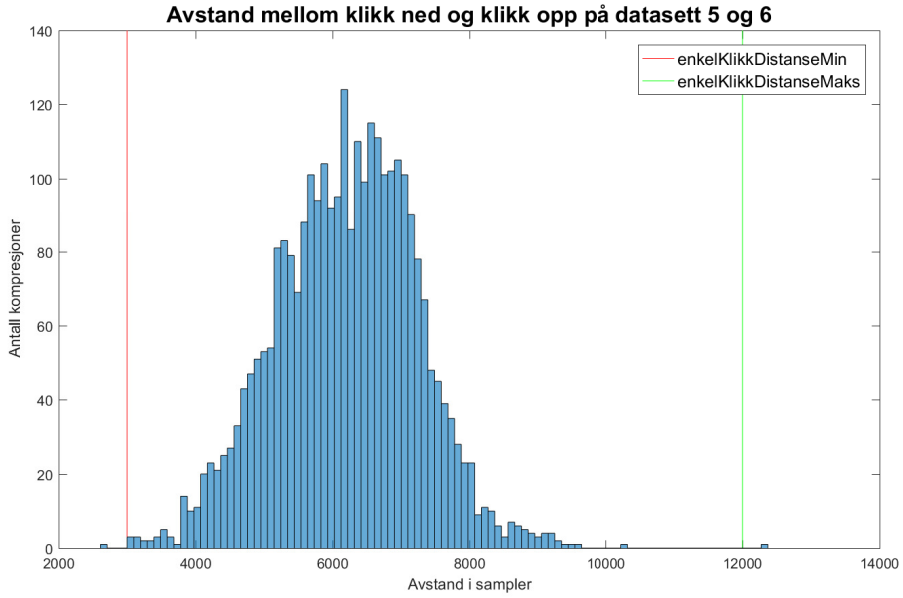
Figur 5.5: Histogrammet viser forholdet mellom amplituden til annoterterte klikk og tilhørende "maksVerdi"

"enkelKlikkDistanseMin" og "enkelKlikkDistanseMaks"

"enkelKlikkDistanseMin" er en parameter som brukes for å angi minste mulige avstand mellom klikk ned og klikk opp i en kompresjon. For å finne denne parameteren er hele datasett 5 og 6 analysert. Avstanden mellom klikk ned og klikk opp for hver eneste kompresjon er

funnet og vist i histogrammet i figur 5.6. Her ser man at de aller fleste kompresjonene har en avstand på mellom 4 000 og 8 000 sampler. For å få med alle kompresjonen er "enkelKlikkDistanseMin" satt til 3 000 sampler. Totalt på datasett 5 og 6 var det kun 1 kompresjon som hadde mindre avstand mellom klikk ned og klikk opp enn 3 000 sampler.

"enkelKlikkDistanseMaks" er en parameter som brukes for å angi største mulige avstand mellom klikk ned og klikk opp i en kompresjon. Denne er valgt til 12 000 sampler. Figur 5.6 viser at 12 000 sampler er godt over den normale avstanden mellom klikk ned og klikk opp.



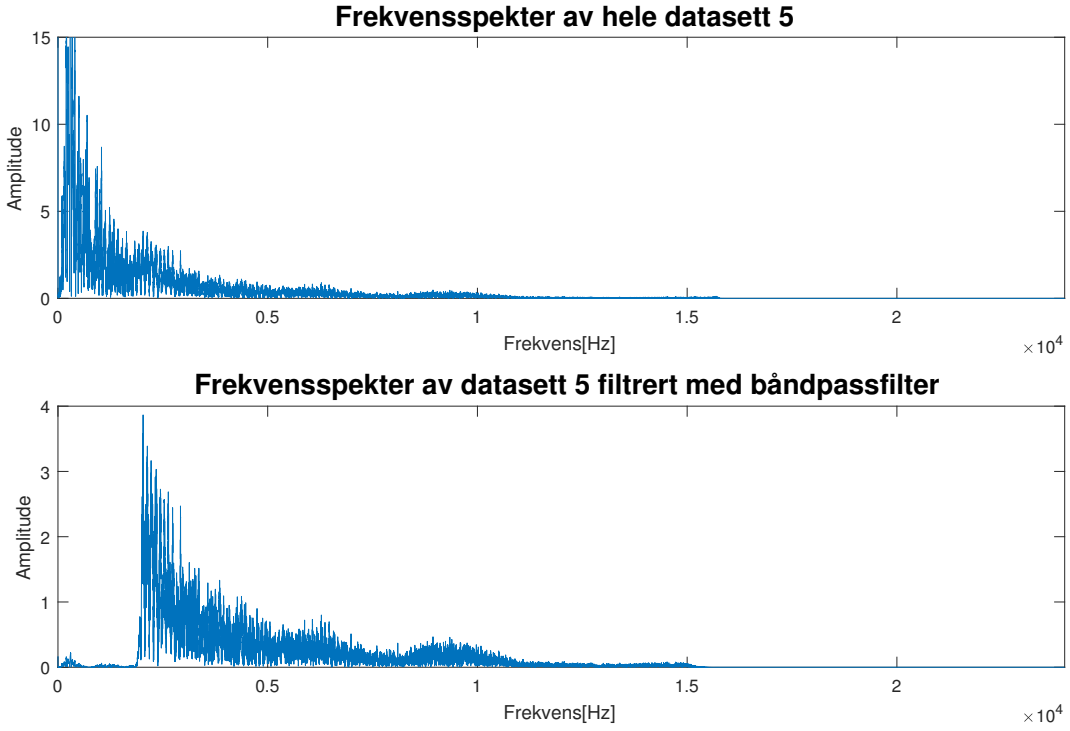
Figur 5.6: Histogrammet viser fordeling av avstand, i sampler, mellom klikk ned og klikk opp. Her er alle kompresjoner fra Datasett 5 og 6 brukt.

"dobbelKlikkDistanse"

"dobbelKlikkDistanse" er en parameter som brukes for å angi minste mulige avstand mellom to kompresjoner. Denne parameteren er satt til 19200 sampler eller 0.4 sekund. Dette tilsvarer en kompresjonsrate på 150 pr. minutt. Når anbefalt kompresjonsrate er 100 kompresjoner pr minutt [22] er det ikke hensiktsmessig å kunne detektere høyere kompresjonsrate enn 150 pr. minutt.

5.5 Eksperiment 5: Valg av båndpassfilter

Kompersjonsdeteksjon skal gjøres i en treningssituasjon hvor mange mennesker, gjerne barn, er samlet i et rom og trener på HLR, se kapittel 1.3. Det er dermed sannsynlig at der kan være mye tale og bakgrunnsstøy i lydopptakene. Dette er nødvendig å filtrere bort for å kunne utføre kompresjonsdeteksjon. Figur 5.7 viser frekvensspekteret av hele datasett 5 funnet ved hjelp av fourier transform. I datasett 5 var det ingen som snakket og man ser at det ikke er noe klart området som klikkelyden har frekvenser innenfor.

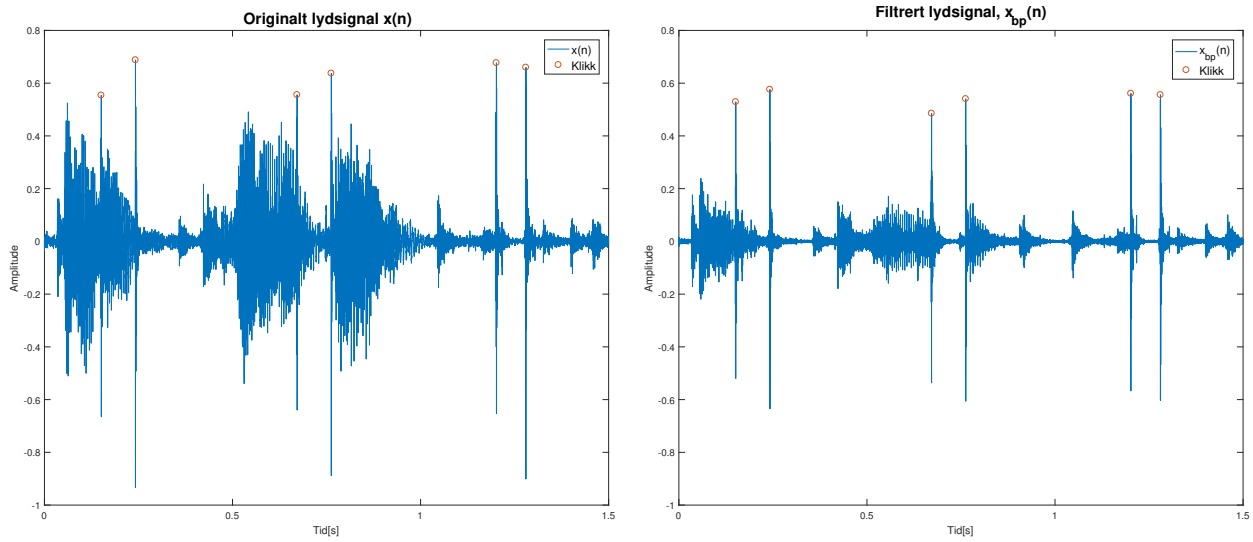


Figur 5.7: Frekvensspekteret av hele datasett 5

Menneskelig tale har en grunnfrekvens på rundt 85-180Hz for menn og 165-255Hz for kvinner [23]. Dette betyr at frekvenser rundt dette område må fjernes for å fjerne talestøy fra lydopptaket. Der er valgt et båndpassfilter som har passbånd mellom 2 000 og 15 000Hz og stoppbånd mindre enn 1 000Hz og mer enn 16 000Hz, se figur 4.3. Stoppbåndet blir dempet med 40dB. Filteret må være så bredt siden klikkelydene har frekvenser over stort sett hele frekvensspekteret. Figur 5.7 viser at over 15 000Hz er det svært lite informasjon. Derfor er det brukt båndpassfilter i stedet for høypassfilter.

Resultat

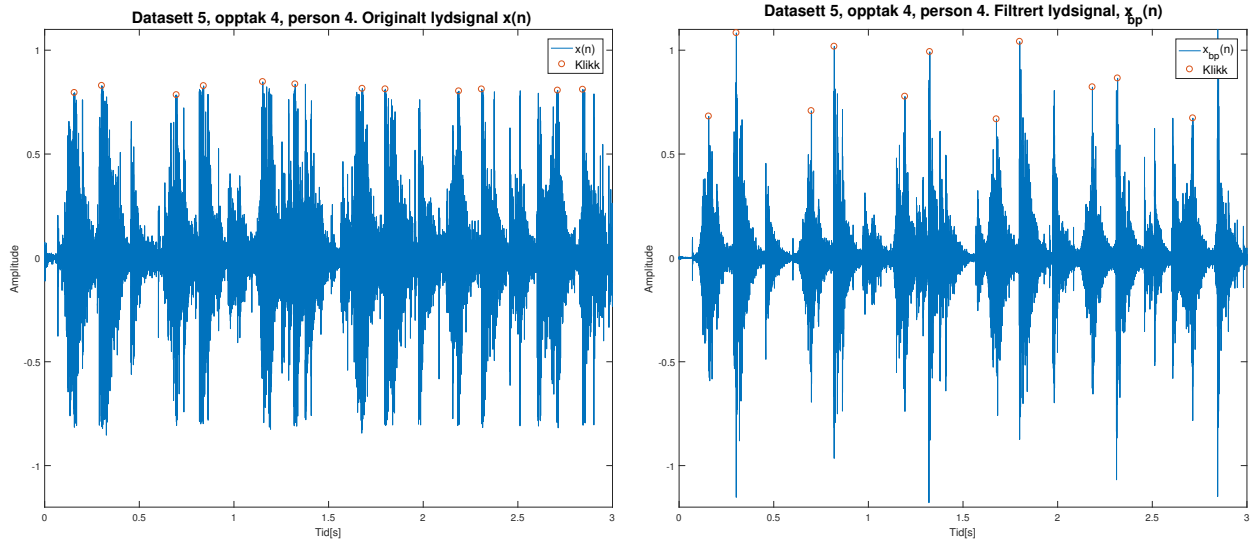
Båndpassfilteret ble testet på opptak 6 i datasett Laerdal. Dette opptaket ble tatt med HLR video i bakgrunnen. Et lite område hvor det var tale i filmen er vist i figur 5.8a. Dette ble filtrert med båndpassfilteret og resultatet er vist i figur 5.8b. Her ser man at filteret virker etter sin hensikt. Støy fra tale ble dempet kraftig, mens klikkelydene blir kun dempet litt. Figur 5.9 er det vist lydopptak fra person 4, opptak 4 i datasett 5. Denne telefonen hadde ekstra sensitiv mikrofon noe som gjorde lydopptaket støyfullt. Båndpassfilteret bidro her til å redusere støyen slik at det blir enklere å detektere kompresjoner.



(a) Ufiltrert lydopptak med talestøy

(b) Lydopptak filtrert med båndpassfilter

Figur 5.8: Viser effekten av å båndpassfiltrere signalet. Her blir talestøy dempet kraftig, mens klikkelyden kun blir dempet litt.



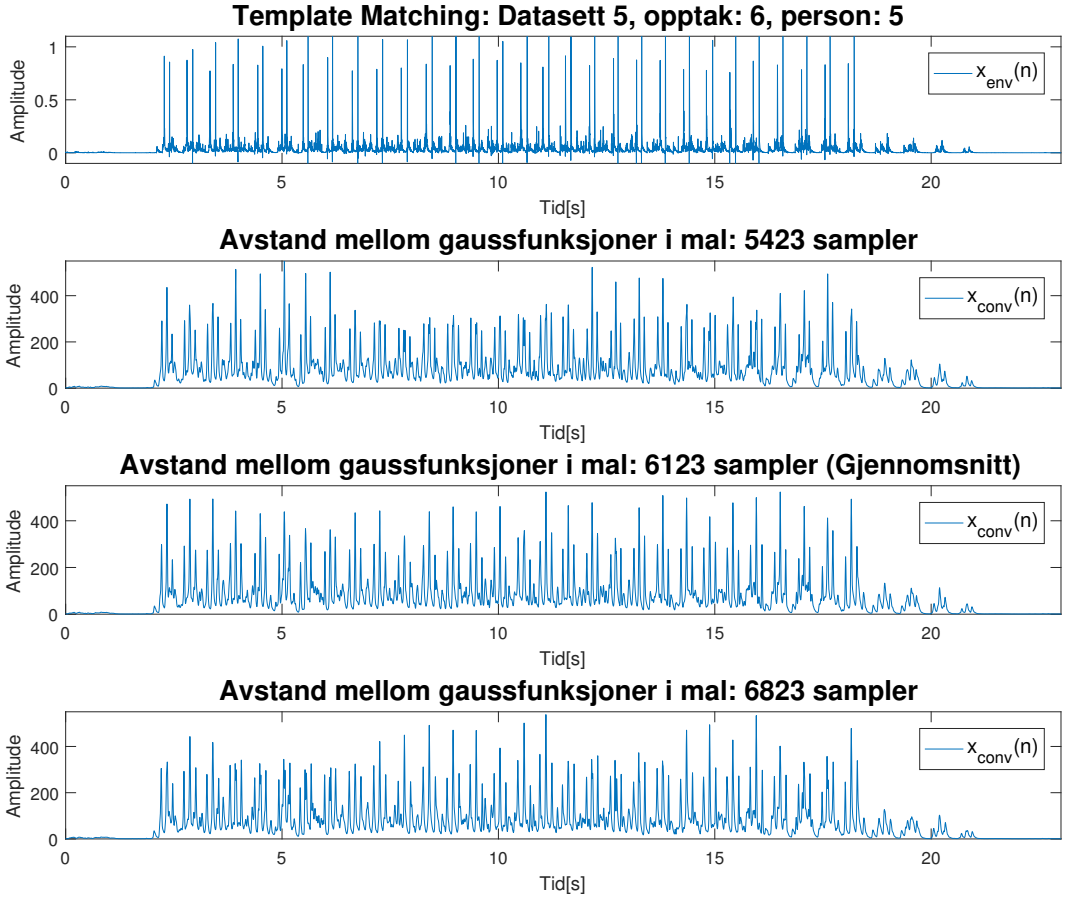
(a) Ufiltrert lydopptak med en telefon som hadde ekstra sensitiv mikrofon.

(b) Lydopptak filtrert med båndpassfilter

Figur 5.9: Figuren viser effekten av å båndpassfiltrere signalet. Her blir en del støy dempet slik at det er lettere å detektere kompresjoner tilhørende dukke 4.

5.6 Eksperiment 6: Valg av avstand mellom gaussfunksjoner i mal

Til dette eksperimentet er det brukt opptak 6, person 5 i datasett 5, se side 95 for protokoll. Under *template matching* konvolveres en mal (template) med $x_{env}(n)$, se kapittel 4.2. Malen er bygget opp av to gaussfunksjoner med en gitt bredde og avstand, se kapittel 4.1.1. Figur 5.12 viser at avstanden mellom klikk ned og klikk opp gjerne varierer med 2 000 sampler i løpet av 30 kompresjoner. For å kompensere for den store variasjon må bredden på gaussfunksjonen være store. Figur 5.6 viser at avstanden mellom klikk ned og klikk opp kan variere mellom 4 000 og 8 000 sampler totalt på datasett 5 og 6. Med så store variasjoner er det ikke mulig å velge én verdi til avstand mellom gaussfunksjonene i malen. En mulig løsning er da å bruke gjennomsnittet av avstandene mellom klikk ned og klikk opp i de fem kompresjonene som er tilgjengelig under initialisering. For å teste hvor mye det har å si med forskjellig avstand mellom gaussfunksjonene er det laget tre maler med avstand 5423 sampler, 6123 sampler (gjennomsnitt) og 6823 sampler. Figur 5.10 viser konvolusjon mellom $x_{env}(n)$ og de tre forskjellige malene.



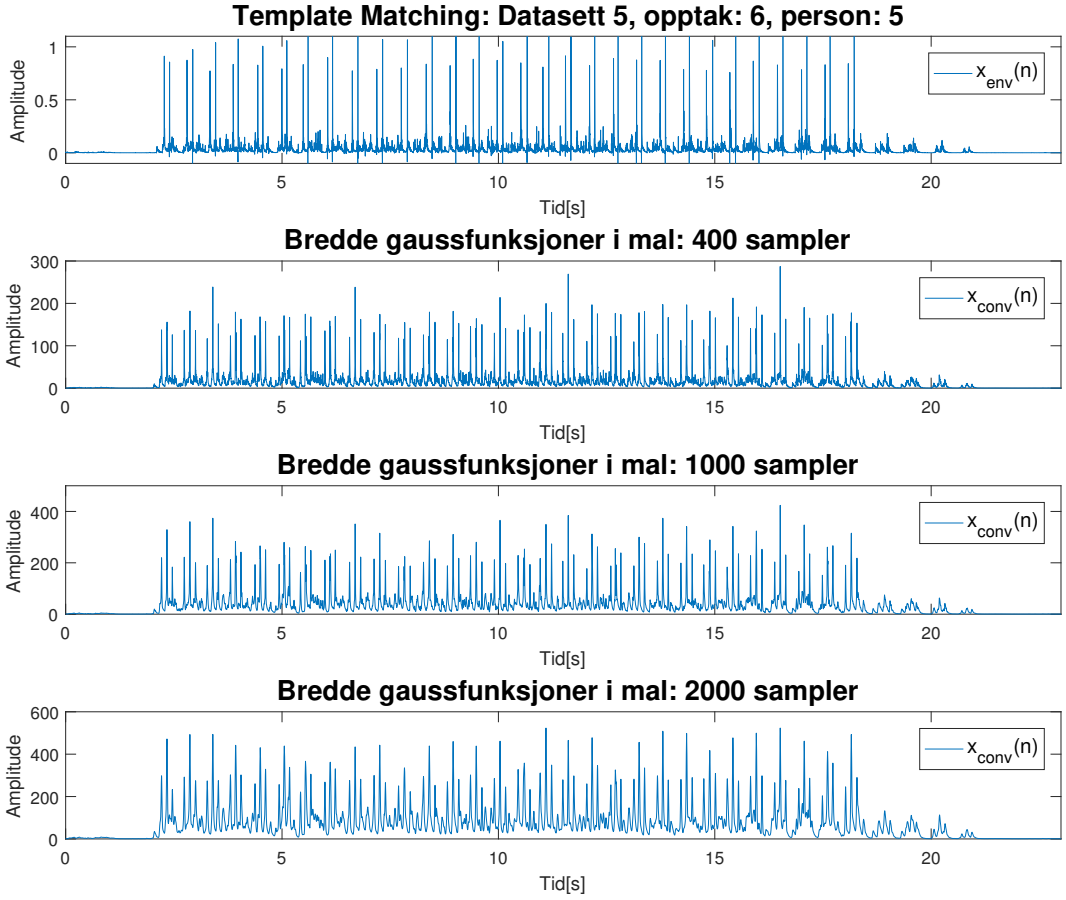
Figur 5.10: Viser konvolusjon mellom $x_{env}(n)$ og mal med forskjellig avstand mellom gaussfunksjonene.

Resultat

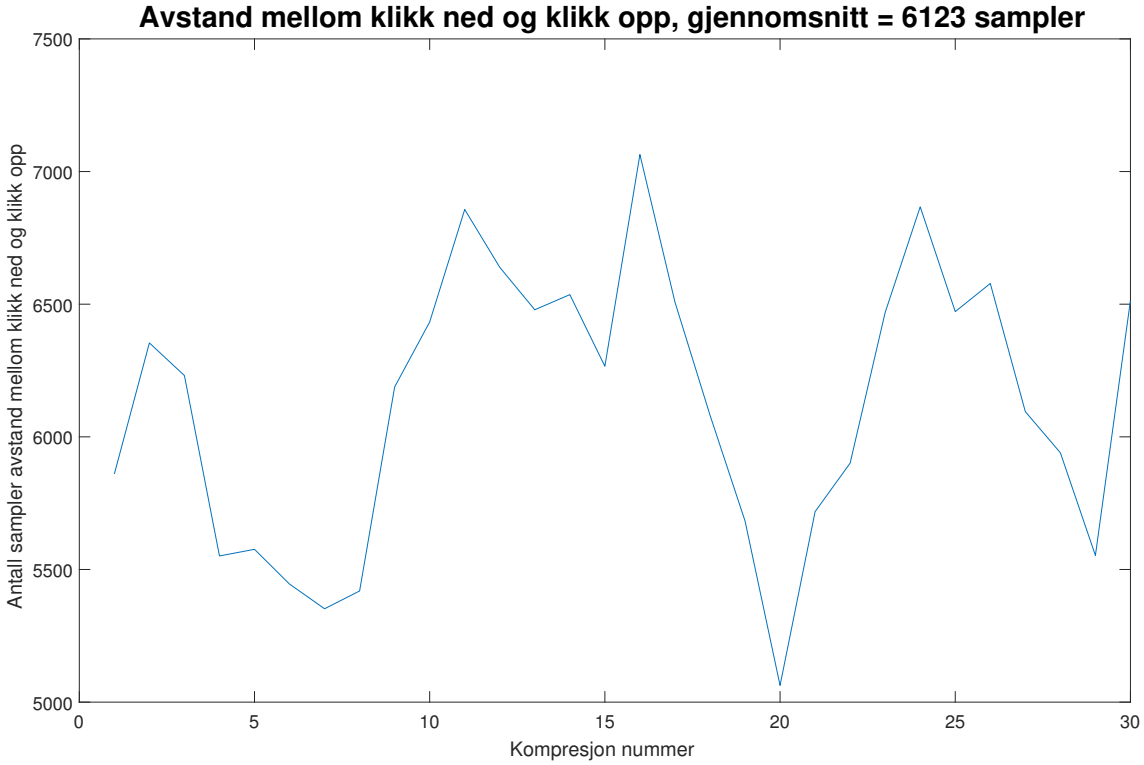
Figur 5.10 viser konvolusjon mellom $x_{env}(n)$ og de tre forskjellige malene. En avstand lik gjennomsnittet av avstandene mellom klikk ned og klikk opp i de fem kompresjonene som er tilgjengelige under initialisering, gir flest topper hvor malen treffer en kompresjon.

5.7 Eksperiment 7: Valg av bredde på gaussfunksjoner i mal

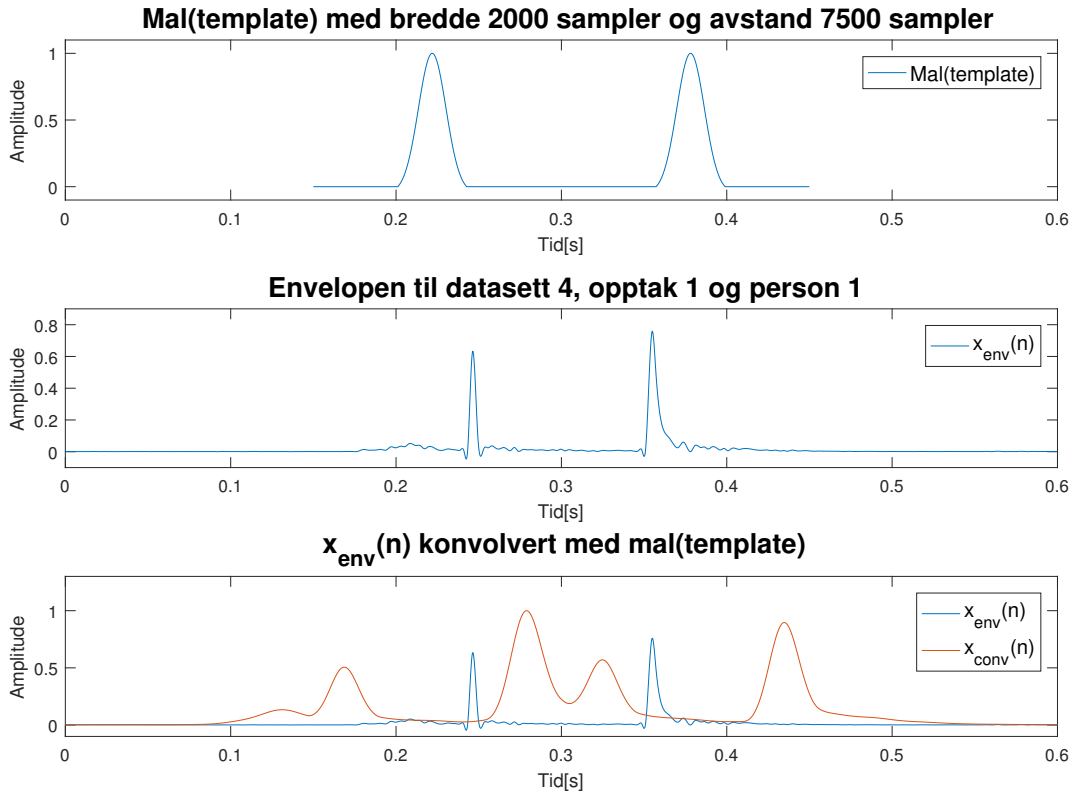
Til dette eksperimentet er det brukt opptak 6, person 5 i datasett 5, se side 95 for protokoll. Malen (template) er bygget opp av to gaussfunksjoner med en gitt bredde og avstand, se kapittel 4.1.1. Avstanden mellom hver gaussfunksjon blir valgt under initialisering, se kapittel 4.1.1. Figur 5.5 viser at det er stor variasjon på avstand mellom klikk ned og klikk opp. Selv på ett enkelt opptak med én person kan avstanden variere med et par tusen sampler, se figur 5.12. Dermed må bredden på gaussfunksjonene være store for å kunne gi god kompresjonsdeteksjon. For å finne en brukbar bredde på gaussfunksjonen i malen ble det testet med tre forskjellige bredder, 400 sampler, 1 000 sampler og 2 000 sampler. Envelopen til et typisk klikk vil ha en bredde på rundt 400 sampler. Derfor er minste bredde på gaussfunksjonen valgt til 400. Konvolusjon mellom de tre malene og $x_{env}(n)$ er vist i figur 5.11.



Figur 5.11: Viser konvolusjon mellom $x_{env}(n)$ og maler med tre forskjellige bredder på gaussfunksjonene.



Figur 5.12: Viser avstand mellom klikk ned og klikk opp for hver kompresjon i Datasett 5, opptak 6 og person 5.



Figur 5.13: Viser problemet som oppstår dersom avstanden mellom gaussfunksjonene i malen er forskjellig fra avstand mellom klikk ned og klikk opp. Begge toppunktene fra malen vil ikke treffe begge toppunktene til $x_{env}(n)$ samtidig. Dermed får man fire toppunkt i stedet for tre hvor det i midten har vesentlig høyere amplitude enn de to på sidene slik figur 2.4 viser.

Resultat

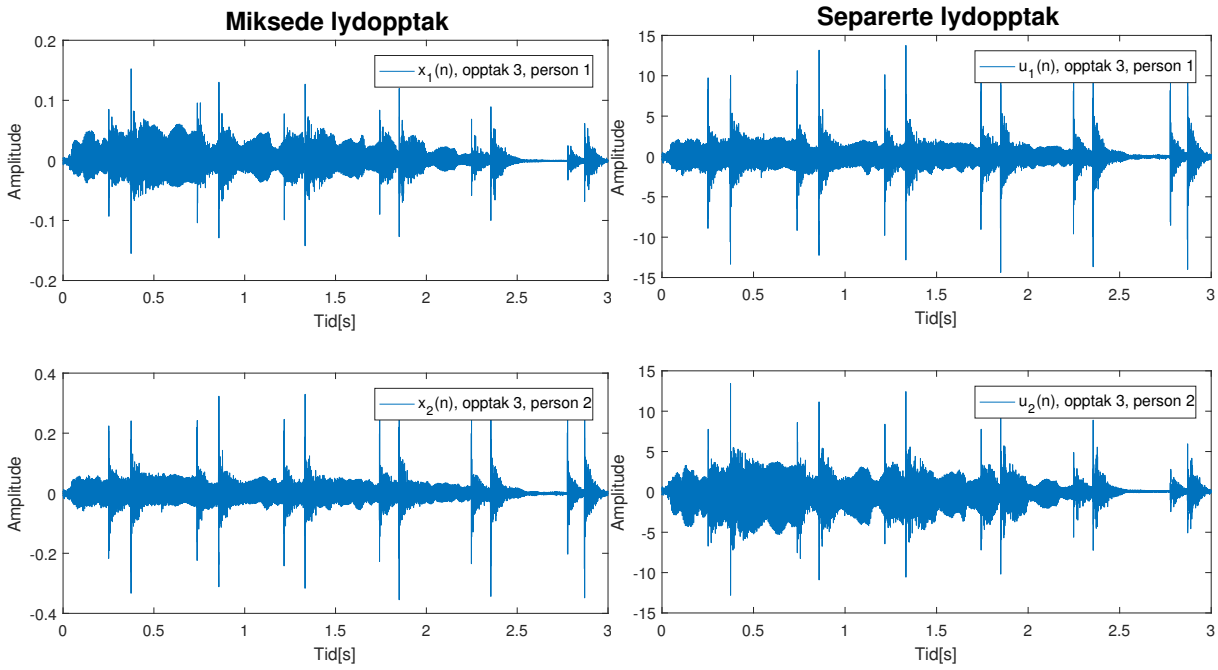
Figur 5.11 viser konvolusjon mellom $x_{env}(n)$ og tre maler med forskjellig bredde. Ved bruk gaussfunksjoner med bredde 400 sampler er det kun noen få kompresjoner som gir en høy amplitude som skiller seg fra resten. Dette kommer av at avstanden mellom toppunktene i malen er en del forskjellig fra avstanden mellom klikk ned og klikk opp, se figur 5.13. Ved å utvide bredden til gaussfunksjonene reduserer man dette problemet. En bredde på 1 000 sampler gir vesentlig flere kompresjoner som gir høy amplitude som skiller seg ut. Ved bruk av en bredde på 2 000 sampler gir alle kompresjonene høyere amplitude i midten hvor det er kompresjon. En bredde på gaussfunksjonene på 2 000 sampler blir dermed anbefalt og brukt videre i initialisering når malen lages.

5.8 Eksperiment 8: Blind Source separation

Eksperiment 8 består av to deler. Eksperiment 8a utfører BSS hvor kildesignalene er mikset i mikrofonene. Eksperiment 8b utfører BSS hvor kildesignalene er mikset lineært på PC.

5.8.1 Eksperiment 8a

Til dette eksperimentet ble det brukt opptak 3 fra datasett 2, se side 90. I opptak 3 ble det både spilt av musikk og komprimert på en MiniAnne dukke slik figur 8.3 viser. Kildesignalene $s_1(n)$ og $s_2(n)$ blir mikset i mikrofonen på hver telefon. De miksede signalene $x_1(n)$ og $x_2(n)$ blir så separert med FastICA [9] algoritmen som er forklart i kapittel 2.1.1. Da får man $u_1(n)$ og $u_2(n)$ som er estimat av kildesignalene $s_1(n)$ og $s_2(n)$.



(a) Miksede lydkilder $x_1(n)$ og $x_2(n)$.

(b) Separerte lydkilder, $u_1(n)$ og $u2(n)$, estimat av kildesignalene $s_1(n)$ og $s_2(n)$

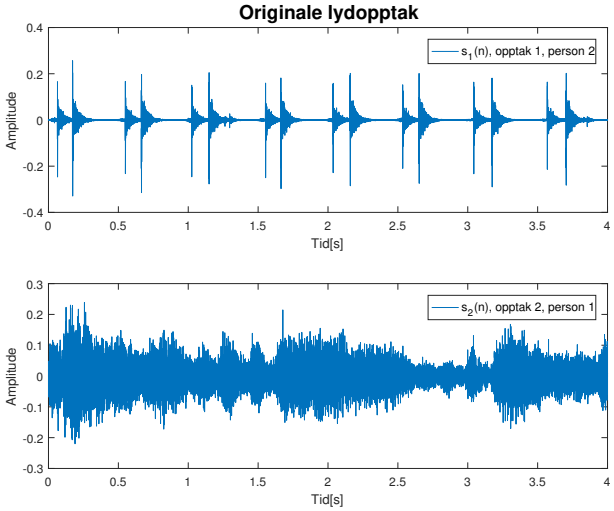
Figur 5.14: Viser to lydkilder fra figur 5.15a som blir mikset digitalt i figur 5.15b og separert i figur 5.15c.

Resultat

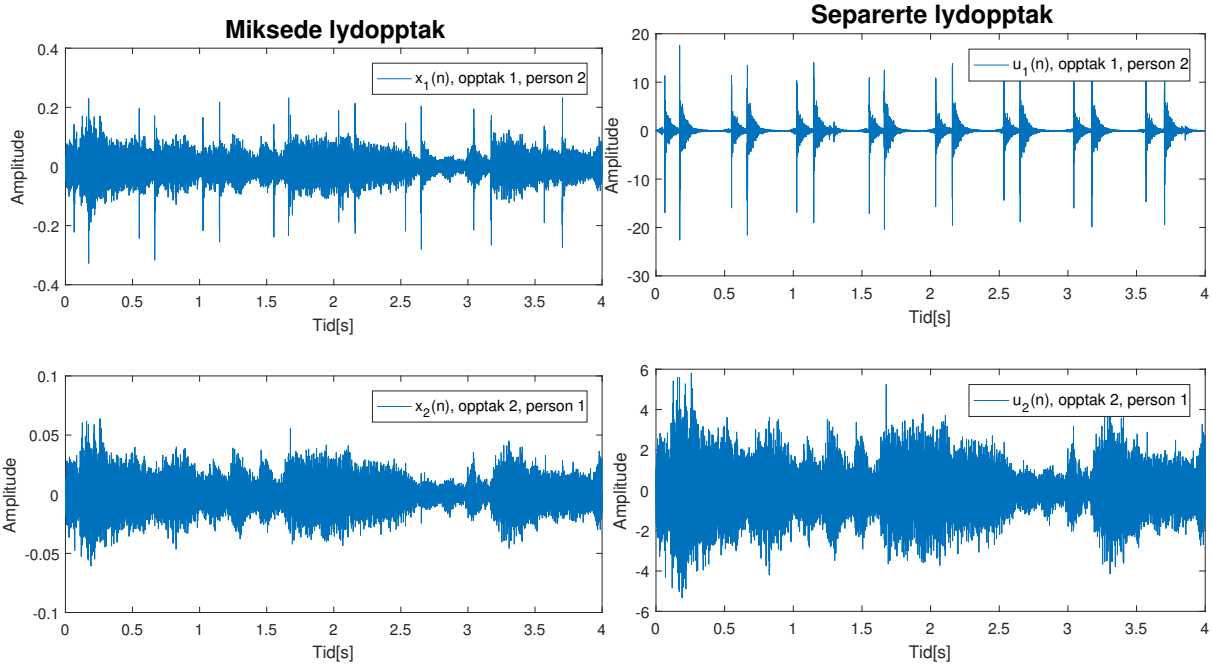
De miksede signalene $x_1(n)$ og $x_2(n)$ er vist i figur 5.15b og $u_1(n)$ og $u_2(n)$ er vist i figur 5.15c. Figur 5.15c viser at det ikke var mulig å skille dukkelyden fra musikken ved hjelp av BSS. Det er doble klikkelyder og musikk på både $u_1(n)$ og $u_2(n)$.

5.8.2 Eksperiment 8b

Til dette eksperimentet ble det brukt opptak 1 og 2 fra datasett 2, se side 90. I opptak 1 ble det kun utført kompresjoner og i opptak 2 ble det kun spilt av musikk. Figur 5.15a viser kildesignalene $s_1(n)$ og $s_2(n)$. Disse kildesignalene ble mikset sammen med lineær miksematrise $\mathbf{A}$ slik ligning 2.3 viser. Miksen av kildesignalene er vist i figur, 5.15b. De miksede signalene $x_1(n)$ og $x_2(n)$ blir så separert med FastICA [9] algoritmen som er forklart i kapittel 2.1.1. Da får man $u_1(n)$ og $u_2(n)$ som er estimat av kildesignalene $s_1(n)$ og $s_2(n)$, se figur 5.15c.



(a) Originale lydkilder $s_1(n)$ og $s_2(n)$.



(b) Miksede lydkilder $x_1(n)$ og $x_2(n)$.

(c) Separerte lydkilder, $u_1(n)$ og $u2(n)$, estimat av kildesignalene $s_1(n)$ og $s_2(n)$

Figur 5.15: Viser to lydkilder fra figur 5.15a som blir mikset digitalt i figur 5.15b og separert i figur 5.15c.

Resultat

Resultatene av eksperiment 8b er vist i figur 5.15. Dersom man sammenligner figur 5.15a og figur 5.15c ser man at $u_1(n)$ ligner mye på $s_1(n)$ og $u_2(n)$ ligner mye på $s_2(n)$. Dette viser at FastICA-algoritmen virker svært bra så lenge kildesignalene er mikset lineært.

5.9 Eksperiment 9: Template Matching

I dette eksperimentet er det utført *template matching* i henhold til metoden beskrevet i kapittel 4.2. Metoden ble testet på datasett 5, datasett 6 samt datasett Laerdal. Det første som gjøres er å lage en mal under initialisering. Denne konvolveres så med ett og ett vindu av lydopptaket. Kompresjonene detekteres så med adaptiv amplitude terskling.

	Datasett 5			Datasett 6			Datasett Laerdal		
tAcc	96.59%			96.79%			96.66%		
PPV	83.45%			86.29%			83.64%		
NPV	98.02%			97.94%			98.02%		
Sen	82.18%			82.22%			81.48%		
Spe	98.19%			98.48%			98.29%		
Forvirrings-matrise		Prediktert			Prediktert			Prediktert	
	Virkelig	P	N	Virkelig	P	N	Virkelig	P	N
	P	978	212	P	1447	313	P	3748	852
	N	194	10515	N	230	14918	N	733	42078

Tabell 5.3: Resultater fra *template matching*

Resultat

Resultatene fra *template matching* er vist i tabell 5.3. Kapittel 2.4 tar for seg forklaring av forvirringsmatrise og de fem ytelsesmålene tAcc, PPV, NPV, Sen og Spe. Resultatene fra *template matching* er relativt gode. Denne metoden fungerte omtrent like bra på alle tre datasett. 82% (Sen) av kompresjonene ble detektert og 83-86% (PPV) av detekterte kompresjoner var innenfor område på $\pm 1\,500$ sampler rundt en annotert kompresjon.

5.10 Eksperiment 10: Adaptiv amplitude terskling

Adaptiv amplitude terskling er den enkleste av metodene vist i figur 4.1. Metoden som er brukt i dette eksperimentet er forklart i kapittel 4.4. Adaptiv amplitude terskling ble testet på datasett 5, datasett 6 og datasett Laerdal.

	Datasett 5	Datasett 6	Datasett Laerdal
tAcc	99.13%	99.95%	96.45%
PPV	95.95%	99.83%	79.48%
NPV	99.49%	99.97%	98.41%
Sen	95.46%	99.72%	85.22%
Spe	99.55%	99.98%	97.65%

Forvirringsmatrise

Virkelig \ Prediktert	P	N
P	1136	54
N	48	10515

Virkelig \ Prediktert	P	N
P	1755	5
N	3	14918

Virkelig \ Prediktert	P	N
P	3920	680
N	1012	42078

Tabell 5.4: Resultater fra Adaptiv Amplitude Terskling

	Datasett 5	Datasett 6	Datasett Laerdal
tAcc	99.48%	99.95%	98.07%
PPV	97.64%	99.83%	87.53%
NPV	99.69%	99.97%	99.29%
Sen	97.23%	99.72%	93.41%
Spe	99.73%	99.98%	98.57%

Forvirringsmatrise

Virkelig \ Prediktert	P	N
P	1157	33
N	28	10515

Virkelig \ Prediktert	P	N
P	1755	5
N	3	14918

Virkelig \ Prediktert	P	N
P	4297	303
N	612	42078

Tabell 5.5: Resultater fra Adaptiv Amplitude Terskling med bruk av annoterte kanaler i stedet for detekterte kanaler. Dersom vi sammenligner med resultatene fra tabell 5.4 ser vi at datasett 5 gjør det litt bedre og datasett Laerdal gjør det mye bedre i forhold til detekterte kanaler. Feil valg av kanaler gjør at resultatene i tabell 5.4 blir dårligere enn i tabell 5.5.

Person/ Opptak	Person 1	Person 2	Person 3	Person 4	Person 5
Opptak 1: H:1.2m L: 0.8m	100%	100%	100%	100%	100%
Opptak 2: H:1.4m L: 0.8m	100%	100%	93.55%	100%	100%
Opptak 3: H:1.6m L: 0.8m	100%	100%	100%	100%	100%
Opptak 4: H:1.2m L: 1.0m	96.55%	100%	16.67%	71.88%	100%
Opptak 5: H:1.2m L: 1.2m	93.1%	100%	100%	70%	100%
Opptak 6: H:1.2m L: 1.4m	100%	100%	96.67%	96.43%	100%
Opptak 7: H:1.4m L: 1.0m	100%	93.1%	100%	100%	100%
Opptak 8: H:1.6m L: 1.2m	100%	96.67%	100%	96.55%	100%

Tabell 5.6: Sensitivitet for hvert opptak i datasett 5. Sensitivitet viser hvor stor andel av kompresjonene som ble detektert.

Person/ Opptak	Person 1	Person 2	Person 3	Person 4	Person 5
Opptak 1: H:1.4m L: 0.8m	100%	93.44%	100%	100%	100%
Opptak 2: H:1.4m L: 1.0m	100%	100%	100%	100%	100%
Opptak 3: H:1.4m L: 1.2m	100%	100%	100%	100%	100%
Opptak 4: H:1.4m L: 1.4m	100%	98.11%	100%	100%	100%
Opptak 5: H:1.4m L: 1.6m	100%	100%	100%	100%	100%
Opptak 6: H:1.4m L: 1.8m	100%	100%	100%	100%	100%

Tabell 5.7: Sensitivitet for hvert opptak i datasett 6. Sensitivitet viser hvor stor andel av kompresjonene som ble detektert.

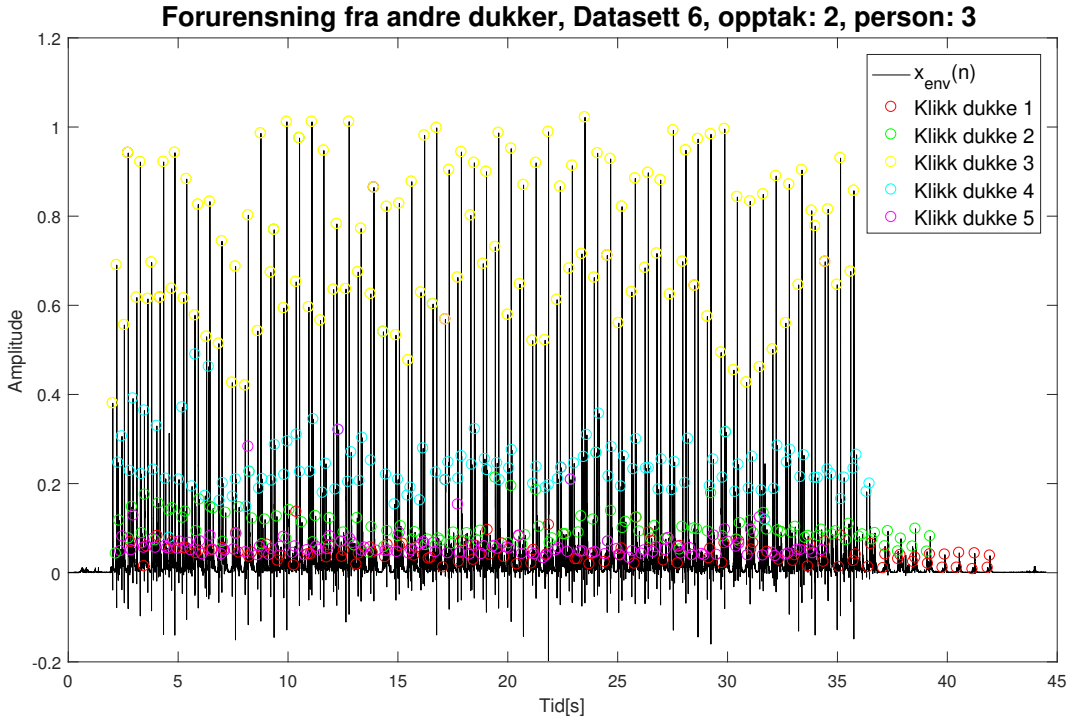
Resultat

Resultatene fra adaptiv amplitude terskling er vist i tabell 5.4 og 5.5. Kapittel 2.4 tar for seg forklaring av forvirringsmatrise og de fem ytelsesmålene tAcc, PPV, NPV, Sen og Spe som brukes i tabell 5.4 og 5.5. Tabell 5.4 viser resultatet for adaptiv amplitude terskling utført på datasett 5, datasett 6 og datasett Laerdal. Resultatene fra adaptiv amplitude terskling er svært gode. I datasett 6 var det kun 3 av 1 760 kompresjoner som ikke ble detektert og kun 5 detekterte kompresjoner som ikke var innenfor området på $\pm 1\,500$ sampler rundt en annotert kompresjon. Datasett 5 gav litt dårligere resultater hvor 95% (Sen) av kompresjonene ble detektert. Med datasett Laerdal ble kun 85% (Sen) av kompresjonene detektert, mens 20% (1-PPV) ble detektert utenfor annoterte kompresjoner. Mye av årsaken til at datasett 5 og datasett Laerdal har dårligere resultater er at noen ganger ble det detektert feil kanaler. I datasett Laerdal ble riktig kanal detektert i 83% av tilfellene mens datasett 5 og datasett 6

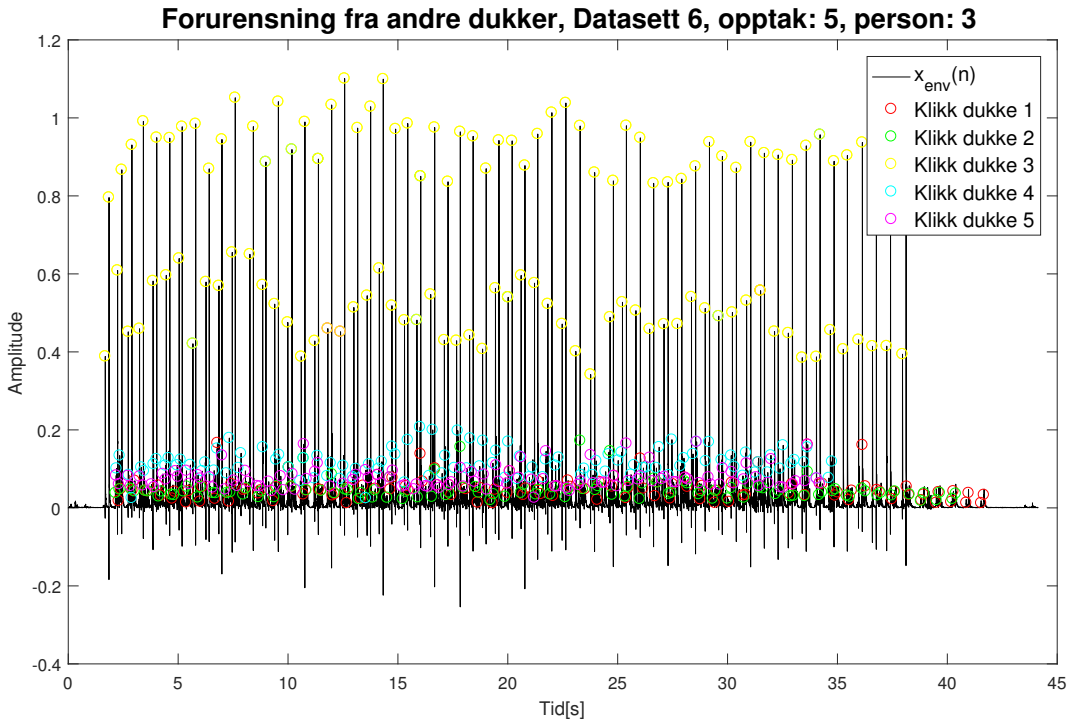
ble riktig kanal detektert i 97.5% og 100% av tilfellene hhv, se tabell 5.1. Tabell 5.5 visere samme resultater fra samme eksperiment bare med bruk av annoterte kanaler i stedet for detekterte kanaler. I tabell 5.6 og 5.7 er sensitiviteten for hvert opptak i datasett 5 og 6 vist. Sensitiviteten viser andel kompresjoner som er detektert. Det er ingen klar sammenheng mellom avstanden mellom personer og andel kompresjoner som er detektert.

5.11 Eksperiment 11: Påvirkning fra nabodukker

For å få tall på hvor mye klikkelydene fra nabodukkene påvirker lydopptakene ble det gjort en avstandstest på datasett 5 og 6, se side 95 og 98 for protokoll. Dette kan brukes for å anbefale en avstand i lengde og høyderetning mellom dukker som gir god deteksjon av kompresjoner, se figur 5.18. Alle opptakene i datasett 5 og 6 er fullstendig annotert med korrekt posisjon for klikk ned, klikk opp og $Det(i)$ midt mellom klikk ned og klikk opp. Opptakene er også synkronisert slik at tid t på lydopptak til person 2 er samme tidspunkt som tid t på lydopptak til person 3. Figur 5.17 viser *envelopen* $x_{env}(n)$ til opptak 5 og person 3 fra datasett 6. Her er det markert hvilke toppunkt som tilhører hvilke dukker. Toppunkt til klikk fra dukke 3 har selvfølgelig høyest amplitude. Det er tatt utgangspunkt i person 3 fordi denne var plassert i midten med personer foran og bak som figur 5.18 viser.



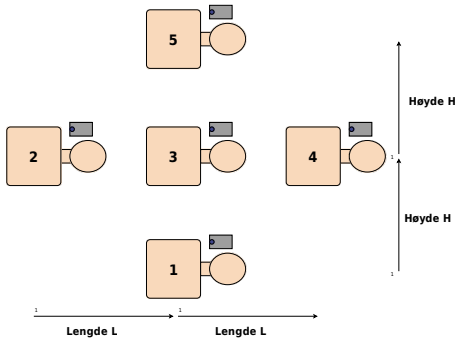
Figur 5.16: Viser envelopen til opptak 2 og person 3 fra datasett 6. Klikk fra hver person er plottet med sirkler. Amplituden på disse er summert for å få et tall på hvilken nabodukke som påvirker opptaket mest.



Figur 5.17: Viser envelopen til opptak 5 og person 3 fra datasett 6. Klikk fra person 1, 2, 4 og 5 er plottet med sirkler. Amplituden på disse er summert for å få et tall på hvilken nabodukke som påvirker opptaket mest.

Lydopptak 5 i figur 5.17 har lavere påvirkning fra nabodukkene enn lydopptak 2 i figur 5.16. Altså er amplituden til klikk fra nabodukker lavere i lydopptak 5 i figur 5.17 enn i lydopptak 2 i figur 5.16 Man ser også at dukke 4 og dukke 2 påvirker lydopptaket til person 3 mest. Person 2 og 4 satt ved siden av person 3, mens person 1 satt bak og person 5 satt foran, se figur 5.18.

Figur 5.18 viser en mulig plassering av dukkene under HLR trening. Denne plasseringen ble brukt i datasett 5 og 6 med forskjellig lengde og høyde for å finne ut hvilken lengde og høyde som ikke gir for stor påvirkningen fra nabodukker. Resultatene fra adaptiv amplitude terskling fra datasett 5 og 6 er

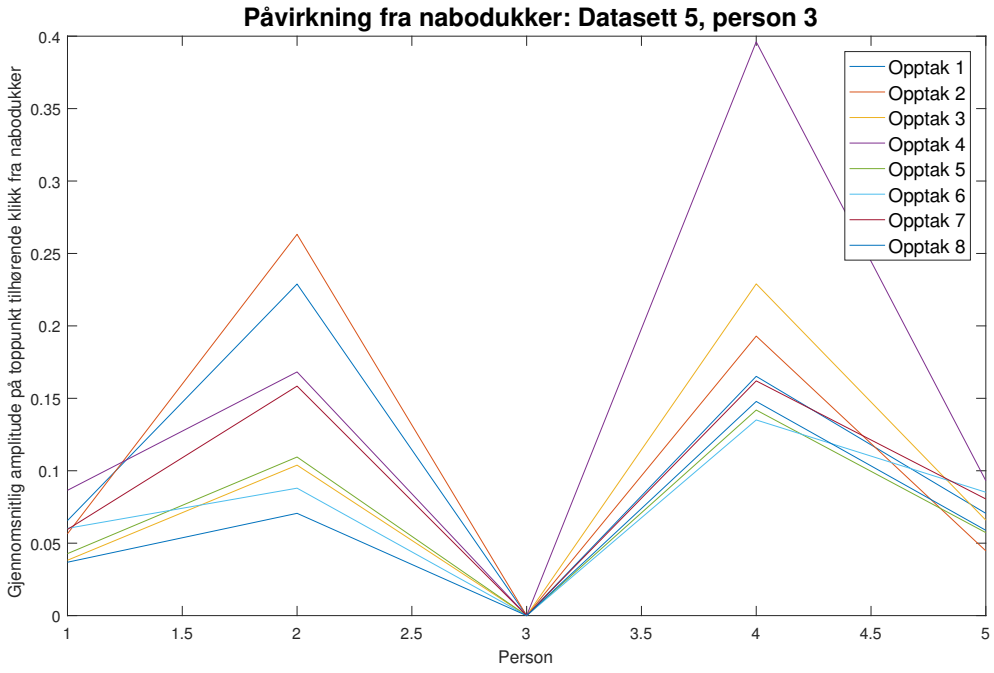


Figur 5.18: Oppsettet til datasett 5 og 6. Viser avstand i lengde og høyderetning.

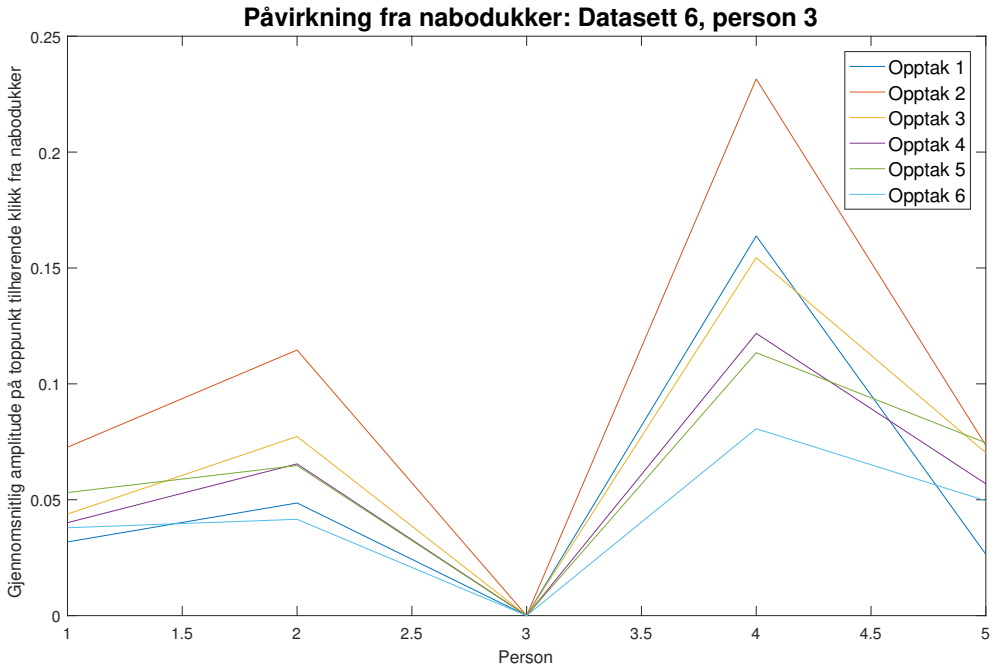
for gode til å kunne brukes for å bestemme avstand, se tabell 5.4. Tabell 5.6 og 5.7 viser at det ikke er en klar sammenheng mellom avstand og andel kompresjoner som ble detektert. Derfor brukes påvirkning fra nabodukker for å bestemme lengden og høyden.

5.11.1 Eksperiment 11a: Valg av høyde

Dette eksperimentet handler om å finne en høyde, slik figur 5.18 viser, hvor påvirkningen fra nabodukker ikke er for stor. Det er brukt datasett 5 og 6 til dette eksperimentet, se side 95 og 98 for protokoll. Figur 5.19 og 5.20 er generert ved å finne gjennomsnittlig amplitude til klikk fra hver dukke på lydopptak fra person 3. Hver graf representerer ett opptak. Hver graf består av 5 punkt, ett for hver person, og tilhørende gjennomsnittlig amplitude på toppunkt tilhørende klikk fra nabodukker.



Figur 5.19: Hver graf viser gjennomsnittlig hvor høy amplitude klikk fra andre dukker fikk på lydopptak fra person 3. Hver graf tilhører ett opptak fra datasett 5.



Figur 5.20: Hver graf viser gjennomsnittlig hvor høy amplitude klikk fra andre dukker fikk på lydopptak fra person 3. Hver graf tilhører ett opptak fra datasett 6.

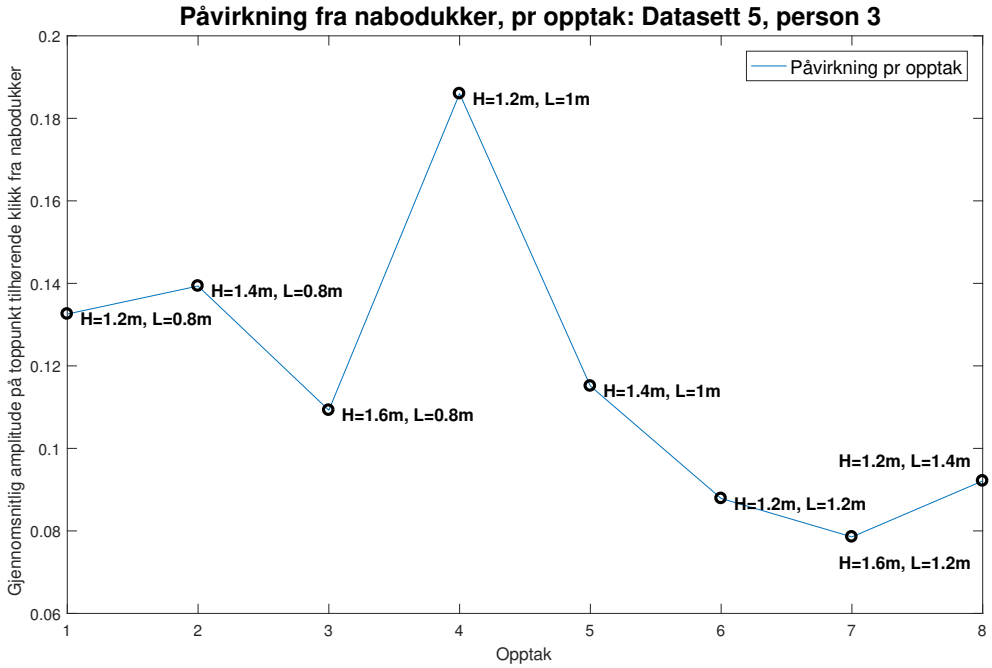
Resultat

Figur 5.19 og 5.20 viser at påvirkning fra sidene (person 2 og 4) er større enn på påvirkning foran og bak (person 1 og 5). Selv på opptak 5 fra datasett 5 og opptak 4 fra datasett 6 hvor avstand i lengde og høyderetning var lik, er det mest påvirkning fra person 2 og 4 fra sidene og minst påvirkning fra person 5 og 1 foran og bak. Gjennomsnittsverdien til "terskelVerdi" på hele datasett 5 er 0.2. Gjennomsnittlig amplitude til klikk fra dukke 1 og 5 på lydopptak fra person 3 er i størrelsesorden 0.05 til 0.1. Derfor betyr det lite om avstanden i høyderetning er 1.2 eller 1.8. Mindre enn 1.2m i høyderetning er litt for trangt for voksne personer og ble derfor ikke testet. Det vil derfor bli anbefalt en høyde på 1.2m eller mer.

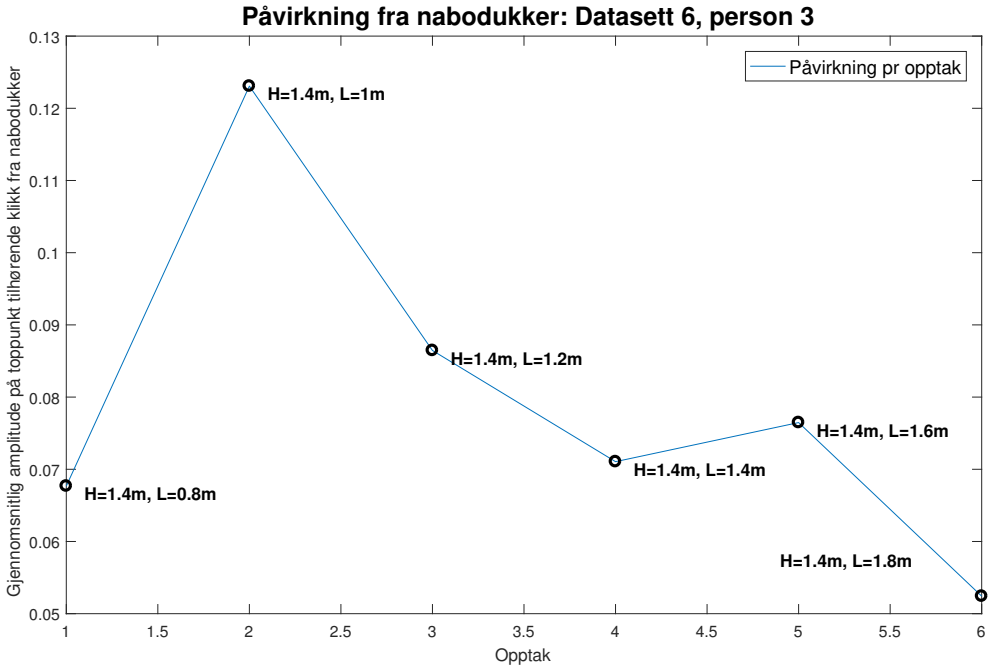
5.11.2 Eksperiment 11b: Valg av lengde

Dette eksperimentet handler om å finne en lengde, slik figur 5.18 viser, hvor påvirkningen fra nabodukker ikke er for stor. Det er brukt datasett 5 og 6 til dette eksperimentet, se side 95 og 98 for protokoll. Figur 5.21 og 5.22 viser gjennomsnittlig amplitude på alle toppunkt tilhørende klikk fra nabodukker fra hvert opptak. Begge figurene er annotert med høyde H

og lengde L mellom dukkene. I figur 5.21 er opptakene sortert i stigende rekkefølge i forhold til lengde L.



Figur 5.21: Viser hvor stor påvirkning klikk fra andre dukker fikk på lydopptaket til person 3 på hvert opptak fra datasett 5. Her er opptakene sortert etter lengde L i denne rekkefølgen: [1 2 3 4 7 5 8 6]



Figur 5.22: Viser hvor stor påvirkning klikk fra andre dukker fikk på lydopptaket til person 3 på hvert opptak fra datasett 6.

Resultat

Av figur 5.21 og 5.22 kan det se ut som det er en sammenheng mellom lengde L og gjennomsnittlig amplitude på toppunkt tilhørende klikk fra alle nabodukkene. Større lengde L gir lavere gjennomsnittlig amplitude på toppunkt tilhørende klikk fra alle nabodukkene. En lengde på minst 1.2m vil bli anbefalt.

Kapittel 6

Diskusjon

I dette kapittelet vil det diskuteres litt mer om hver av metodene.

6.1 Template matching

I eksperiment 9 ble det utført eksperiment med *template matching*. Dette gav relativt gode resultater hvor rundt 82% av kompresjonene ble detektert. Eksperiment 6 og 7 viste at utforming av mal (template) har mye å si for muligheten til å detektere kompresjoner. En mal bestod av to gaussfunksjoner med fast bredde og variabel avstand. Mye av grunnen til at kun 82% av kompresjonene ble detektert var at avstanden mellom klikk ned og klikk opp varierte for mye. Avstanden kunne gjerne variere mellom 5 000 og 7 000 sampler, se figur 5.12. Derfor er *template matching* ikke anbefalt til å detektere kompresjoner.

6.2 Blind source separation

I eksperiment 8 ble det utført eksperiment med *blind source separation* (BSS). Det fungerte svært bra å separere kildesignalene når de var mikset digitalt på pc. Verre var det når kildesignalene ble mikset i mikrofonene, altså at det ble tatt opp lyd med to kildesignaler (klikkelyder og musikk) og to mikrofoner (to telefoner). Figur 5.14 viser resultat fra når kildesignalene ble mikset i mikrofonene. Her er det både musikk og kompresjoner på separerte signaler. Følgende kan være noen av årsakene til at separasjonen ikke ble vellykket:

- Det kan være lyden ikke er helt 100% riktig synkronisert. Dersom det ene lydopptaket

er forskjøvet noen sampler i tid i forhold til det andre vil separasjonen ikke bli vellykket.

- Lydkildene er ikke mikset lineært. Dersom lydkildene ikke er mikset lineært vil det ikke være mulig å skille $s_1(n)$ fra $s_2(n)$, da signalet fra mikrofonene ble modellert som et lineært system.
- Forskjellige mikrofoner eller prosessering av lydsignalet.

Det øverste punktet er det mulig å gjøre noe med dersom det kun er synkronisering som er problemet. Punktet i midten kan kanskje løses med annen algoritme for BSS. Det nederste punktet kan det ikke gjøres noe med. På grunn av dette blir ikke *Blind source separation* anbefalt til bruk ved deteksjon av kompresjoner.

6.3 Adaptiv amplitude terskling

I eksperiment 10 ble det utført eksperiment med adaptiv amplitude terskling. Denne algoritmen gav svært bra kompresjonsdeteksjon hvor andel detekterte kompresjoner var på 99.7%, 95.5% og 85.2% på henholdsvis datasett 6, 5 og Laerdal. Den største utfordringen i denne metoden er egentlig å detektere riktig kanal på lydopptaket. Av og til er det slik at klikk fra nabodukken gir høyest amplitude på den ene kanalen, mens klikk fra nærmeste dukke gir høyest amplitude på den andre kanalen. Dersom det da blir valgt feil kanal detekteres kompresjoner på nabodukken i stedet for tilhørende dukke. Ved bruk av annoterte kanaler i stedet for detekterte kanaler ble andel detekterte kompresjoner på 99.7%, 97.2% og 93.4% på henholdsvis datasett 6, 5 og Laerdal. Totalt sett gav denne metoden best resultater og var i tillegg minst komplisert og ressurskrevende. Dermed blir metoden adaptiv amplitude terskling anbefalt til å detektere kompresjoner.

6.4 Avstand mellom personer under HLR-trening

Plassering til MiniAnne-dukkene som er brukt i datasett 5 og 6 er vist i figur 8.5. Her er det markert avstand mellom dukkene i form av lengde L og høyde H. Tilsvarende plassering ble også brukt i datasett Laerdal, da med 3x3 personer. Det var ikke mulig å anbefale en avstand ut fra andel detekterte kompresjoner. Tabell 5.6 og 5.7 viser at det ikke er en klar sammenheng mellom avstand i lengde og høyderetning og andel kompresjoner detektert.

Derfor ble det i eksperiment 11 gjort en test for å finne ut hvor mye påvirkning lydopptaket fikk fra nabodukkene. Altså hvor høy amplituden til klikk fra nabodukkene er på lydopptaket. Det ble tatt utgangspunkt i lydopptak fra person 3 da denne hadde plassering i midten, se figur 8.5. Fra figur 5.19 og 5.20 så det ut til at lydopptaket ble mest påvirket av dukker fra sidene fremfor dukker foran og bak. Dette kan være fordi personene som komprimerer "skygger" for lyden i større grad foran og bak enn på sidene. En avstand i høyderetning kan dermed være ganske liten. Det ble ikke testet mindre høyde H enn 1.2m da dette var ganske trangt for voksne personer. En høyde H på 1.2m eller større blir dermed anbefalt.

I figur 5.21 og 5.22 ser det ut til å være en sammenheng mellom avstand i lengderetning og gjennomsnittlig amplitude på klikk fra nabodukker slik figur viser. Det ble basert på disse figurene anbefalt en lengde på 1.2m eller større. Ved implementasjon av algoritmen i en app kan det gjøres nye tester for å finne ut hvilke avstander som fungerer best.

Kapittel 7

Konklusjon og videre arbeid

7.1 Konklusjon

Oppgavens mål var å lage en algoritme for deteksjon av kompresjoner og finne ut om dette lar seg gjøre i praksis. Det er testet tre forskjellige algoritmer for deteksjon av kompresjoner. Den beste algoritmen var adaptiv amplitude terskling hvor andel kompresjoner detektert var på 99.7%, 95.5% og 85.2% på henholdsvis datasett 6, 5 og Laerdal. Som konklusjon kan man si at at deteksjon av kompresjoner lar seg gjøre i praksis og metoden adaptiv amplitude terskling gir gode resultater.

7.2 Videre arbeid

Et naturlig neste steg i videre arbeid er å implementere algoritmen adaptiv amplitude terskling i en app på smarttelefon. Dette kan så testes for å finne ut hvor bra deteksjonen fungerer og om endringer må gjøres. Initialiseringen må muligens fjernes fra algoritmen for å gjøre den mer brukervennlig. Det kan være litt vanskelig for barn som ikke har trent på HLR før å komprimere dypt nok fem ganger i starten.

Bibliografi

[1] K. Lexow. http://legeforeningen.no/PageFiles/7128/Norsk%20Resuscitasjonsr%C3%A5d%20(NRR).pdf, 2008.

[2] T. M. Gustavsen. http://www.vegvesen.no/om+statens+vegvesen/presse/nyheter/nasjonalt/135-omkom-i-2016-trafikken, 2017.

[3] I. Hasselqvist-Ax, G. Riva, J. Herlitz, M. Rosenqvist, J. Hollenberg, P. Nordberg, M. Ringh, M. Jonsson, C. Axelsson, J. Lindqvist, *et al.*, "Early cardiopulmonary resuscitation in out-of-hospital cardiac arrest," *New England Journal of Medicine*, vol. 372, no. 24, pp. 2307–2315, 2015.

[4] K. Sunde, K. O. Fremstad, J. Furuheim, and P. A. Steen, "Utrykningstid for ambulansetjenesten i oslo ved hjertestans," *TIDSSKRIFT-NORSKE LAEGEFORENING*, vol. 121, no. 8, pp. 900–903, 2001.

[5] NRR, "Retningslinjer 2015, hlr med hjertestarter, hlr for helsepersonell," 2015. [Online; accessed 28-May-2017].

[6] E. C. Cherry, "Some experiments on the recognition of speech, with one and with two ears," *THE JOURNAL OF THE ACOUSTICAL SOCIETY OF AMERICA*, vol. 25, no. 5, 1953.

[7] A. Westner and V. M. Bove Jr, "Applying blind source separation and deconvolution to real-world acoustic environments," in *Audio Engineering Society Convention 106*, Audio Engineering Society, 1999.

[8] N. Soldati, V. D. Calhoun, L. Bruzzone, and J. Jovicich, "Ica analysis of fmri with real-time constraints: an evaluation of fast detection performance as function of

algorithms, parameters and a priori conditions," *Frontiers in human neuroscience*, vol. 7, p. 19, 2013.

[9] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4, pp. 411–430, 2000.

[10] H. Aapo, K. Juha, and O. Erkki, *Independent component analysis*. 2001.

[11] C. Gari, F. John, and G. Julie, "Biomedical signal and image processing (pdf)," 2005.

[12] J. V. Stone, *Independent component analysis*. Wiley Online Library, 2004.

[13] Wikipedia, "Central limit theorem — wikipedia, the free encyclopedia," 2017. [Online; accessed 5-May-2017].

[14] Wikipedia, "Negentropy — wikipedia, the free encyclopedia," 2017. [Online; accessed 21-May-2017].

[15] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[16] T. Kyriacou, G. Bugmann, and S. Lauria, "Vision-based urban navigation procedures for verbally instructed robots," *Robotics and Autonomous Systems*, vol. 51, no. 1, pp. 69–80, 2005.

[17] Y. Lee, T. Hara, H. Fujita, S. Itoh, and T. Ishigaki, "Automated detection of pulmonary nodules in helical ct images based on an improved template-matching technique," *IEEE Transactions on medical imaging*, vol. 20, no. 7, pp. 595–604, 2001.

[18] K. Yoshii, M. Goto, and H. G. Okuno, "Automatic drum sound description for real-world music using template adaptation and matching methods.," in *ISMIR*, pp. 184–191, 2004.

[19] J. W. A. S. A. G. K. C. Richard Johnson, *Software Receiver Design: Build Your Own Digital Communication System in Five Easy Steps*. Cambridge University Press, 2011.

[20] Wikipédia, "Transformada de hilbert — wikipédia, a enciclopédia livre," 2015. [Online; accessed 31-outubro-2015].

[21] Wikipedia, "Confusion matrix — wikipedia, the free encyclopedia," 2017. [Online; accessed 31-May-2017].

[22] J. Å. Olsen and F. S. Stecher, "Retrospektiv analyse av hjerte-lungeredning (hlr) utført av ambulansepersonell i oslo ved hjelp av transthorakal impedansemåling.," Master's thesis, 2006.

[23] Wikipedia, "Voice frequency — wikipedia, the free encyclopedia," 2016. [Online; accessed 24-May-2017].

Kapittel 8

Vedlegg

8.1 Protokoller

8.1.1 Laerdal

Deteksjon av kompresjon ved bruk av lyd og video fra smarttelefon under HLR

Testansvarlig: Øyvind Stensland

Dato: 18.01.2017

Sted: Laerdal Medical, Tanke Svilandsgate 30

Protokoll for Datasett Laerdal:

Bakgrunn:

Ved trening på hjerte – lungeredning (HLR) brukes ofte en dukke kalt «Mini Anne». Denne lager klikkelyder ved korrekte kompresjoner. I et UiS – Lærdal samarbeid skal det utvikles en løsning for å detektere «klikk» lyder fra «Anna dukken» ved bruk av smarttelefon for å gi tilbakemelding til hver person (og veileder) hvor bra HLR gjennomføres. Dette er enkelt å få til dersom kun en dukke er i bruk. Men i HLR trening er det gjerne 20-50 personer samlet i et klasserom. Da er det ikke like lett å skille dukke fra dukke ved bruk av lydopptaket.

I en virkelig situasjon hvor for eksempel en skoleklasse trener på HLR ved bruk av «Mini Anne» og vår fremtidige applikasjon vil noe variere mens noe er tilnærmet konstant/ standard. Dette er gjengitt i Tabell 1:

Variable:	**Standard:**
Dybde	Plassering av telefon
Rate	Mini Anne (Tilnærmet like dukker/ like lyder)
Hvem som trykker	Samme video kjøres i bakgrunnen
Antall kompresjoner	
Mobiltelefon (Mic, Hz)	
Avstand mellom dukkene	

Tabell 1 Viser hva som vil variere og hva som vil være konstant ved fremtidig bruk av applikasjonen

Ved bruk av denne protokollen får vi testet forskjellig variable slik vi får simulert flere tenkte virkelige situasjoner.

Informasjon og forarbeid for hver deltaker:

- Sørg for å ha minst 1GB ledig plass på telefonen. Dette for å slippe å overføre underveis.
- Sørg for å ha nok strøm på mobiltelefonen.
- Gjerne sett deg litt inn i bruk av mobilkamera. – Hvordan man stiller oppløsning og bruker frontkamera. Ønsket oppløsning er VGA(480p) for å hindre å oppta for mye plass på telefonen. Full HD (1080p) som er vanlig opptar 4 ganger så mye plass som VGA.
- Når videoopptakene skal leveres endres navnet på filene til opptaksnummeret. (fra 01 til 13). Bruk klokkeslettene som er skrevet ned underveis for å finne ut hvilket opptak de hører

sammen med. Filene lagres her: T:/ UiS MiniAnne clicker/[test-person]/ hvor "[test-person]" erstattes av et nummer fra 01 til 09 avhengig av hvilken testperson man er.

Forarbeid for testansvarlig:

- Lage underlag/mal med silhuett av dukke og mobiltelefon, for å sikre standard plassering av dukke og mobiltelefon.
- Skaffe 9 MiniAnne-dukker (Laerdal)
- Nummerere dukkene
- Skaffe video + avspillingsmulighet
- Målbånd
- Stoppeklokke
- Forberede informasjon til deltakere
- Tilrettelegging for overføring av video-/lydfiler
- Lage dropboxmappe el.l. slik at jeg kan få tilgang til filene
- Skrive ut sidene med tester i ni eksemplarer.

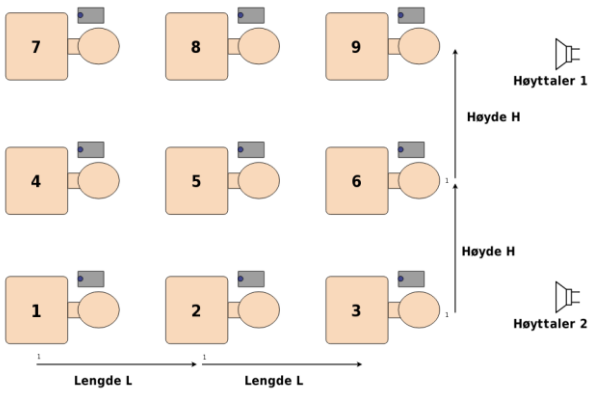
Figur 1 Innsamling av datsett Laerdal hos Laerdal Medical AS

Testoppsett:

Mål: å gjøre video-opptak med telefonens frontkamera, med telefonen plassert slik som på Figur 2 med mikrofonen pekende mot venstre.

Figur 2 Oppsett for en dukke som viser plassering av telefon

Dukkene vil bli plassert slik som på Figur 3. Det vil være konstant avstand mellom dukkene. Dukkene er ca. 50cm lange slik at L kan ikke være mindre enn det. For å ha plass til å gjøre kompresjoner må H være mer enn 1m. Et bilde som viser innsamling av datasett Laerdal er vist i Figur 1. Dette er med L = 1.2m og H = 1.8m.



Figur 3 Plassering av alle dukkene og høyttaler

Alle 13 opptak filmes av alle testpersoner samtidig. Dukkene skal være innstilt på barn. For å ha samme tidspunkt på alle videoopptakene vil vi enten skru av og på lyset, eller klappe, i begynnelsen av hvert opptak. Dermed kan jeg klippe alle filmene i ettertid slik at de har lik tidsakse.

Opptak:

For å lettere kunne skille mellom videoopptakene skrives de neste sidene ut, underveis fyller hver deltager inn tidspunktet hver video ble tatt. Sjekk tidspunkt etterpå.

Opptaksliste for test-person nr. _____:

Test 1

Hensikt med testen:

Teste hvordan avstand mellom hver dukke påvirker muligheten til å skille dukke fra dukke.

Testmiljø: Møterom med teppe.

Opptakene på denne siden gjøres ved at alle personer gjør det samme. Her varierer avstanden mellom dukkene.

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall kompresjoner:**	**Notat:**	**Klokkeslett:**
01	0.7m	1.2m	100/min	Normal	30	Test avstand	
02	0.7m	1.8m	100/min	Normal	30	Test avstand	
03	1.2m	1.8m	100/min	Normal	30	Test avstand	

Test 2

Hensikt med testen:

Teste hvor mye amplituden på klikkelyden dempes i hver posisjon.

Testmiljø: Møterom med teppe.

I opptak 04 testes dukkene i rekkefølge. Man starter med person 1 og ender med person 9. Hver person klikker 20 ganger hver. I opptak 05 komprimerer alle utenom en og en person. Totalt varer dette i 90 sekund. Person 1 har pause fra 0-10, person 2 har pause fra 10-20. Person 9 har pause fra 80 til 90.

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall:**	**Notat:**	**Klokkeslett:**
04	1.2m	1.8m	100/min	Normal	20 klikk	En og en komprimerer 20 klikk hver.	
05	1.2m	1.8m	100/min	Normal	90 sekund	Alle utenom en komprimerer.	

Test 3

Hensikt med testen:

Finne ut hvor lett det er å skille klikkelyder fra filmen fra klikkelydene fra hver dukke.

Testmiljø: Møterom med teppe.

I test 3 skal alle gjøre det samme samtidig.

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall klikk:**	**Notat:**	**Klokkeslett:**
06	1.2m	1.8m	100/min	Normal	40	Video i bakgrunn (7:20)	

Test 4

Hensikt med testen:

Dersom man komprimerer hardt kommer det merkelige «pustelyder» fra dukken. Målet er å finne ut hvordan disse lydene påvirker mulighet til å detektere klikk. I tillegg øker muligens amplituden dersom man trykker hardt. Dette vil man teste i opptak 08.

Testmiljø: Møterom med teppe.

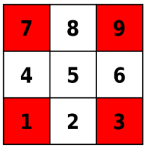
I opptak 07 skal alle komprimere hardt for å fremprovosere lyder som ikke kommer under vanlig bruk.

I opptak 08 skal gruppe 1 komprimere så dypt de klarer mens gruppe 2 komprimerer slik at det akkurat kommer klikkelyd.

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall klikk:**	**Notat:**	**Klokkeslett:**
07	1.2m	1.8m	100/min	Dypt/Hardt	40	Test dybde	

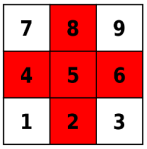
Gruppe 1: person 1, 3, 7, 9

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall klikk:**	**Notat:**	**Klokkeslett:**
08	1.2m	1.8m	100/min	Dypt/ hardt	30	Test dybde	



Gruppe 2: person 2, 4, 5, 6, 8

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall klikk:**	**Notat:**	**Klokkeslett:**
08	1.2m	1.8m	100/min	Normal	30	Test dybde	



Test 5

Hensikt med testen:

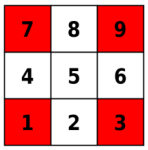
Teste forskjellig kompresjonsrater samtidig

Testmiljø: Møterom med teppe.

I test 5 komprimerer gruppe 1 og gruppe 2 med forskjellig rate.

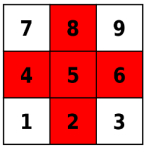
Gruppe 1: person 1, 3, 7, 9

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall sekund:**	**Notat:**	**Klokkeslett:**
09	1.2m	1.8m	100/min	Normal	30	Test raten	
10	1.2m	1.8m	80/min	Normal	30	Test raten	



Gruppe 2: person 2, 4, 5, 6, 8

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall sekund:**	**Notat:**	**Klokkeslett:**
09	1.2m	1.8m	120/min	Normal	30	Test raten	
10	1.2m	1.8m	120/min	Normal	30	Test raten	



Test 6

Hensikt med testen:

Hensikten med denne testen er å teste hvor godt lyden fra dukken kan skilles fra klikkelydene.

Testmiljø: Møterom uten teppe.

Test 7 foregår på et møterom uten teppe hvor det blir en del ekstra støy. I opptak 11 komprimerer alle normalt, mens i opptak 12 komprimerer alle hardt. Opptak 13 foregår slik at en og en komprimerer 10 ganger hvor person nummer 1 starter og person nummer 9 slutter.

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall klikk:**	**Notat:**	**Klokkeslett:**
11	1.2m	1.8m	100/min	Normal	40	Test dårlig underlag	
12	1.2m	1.8m	100/min	Dypt/ hardt	40	Test dårlig underlag	
13	1.2m	1.8m	100/min	Normal	10 klikk hver, en og en	Test dårlig underlag	

Test 7

Hensikt med testen:

Hensikten med denne testen er å teste hvor mye klikkelyder fra dukkene rundt påvirker opptaket.

Testmiljø: Møterom med teppe.

Person 1, 3, 7 og 9 komprimerer 30 ganger med klikkelyd på alle.

Person 2, 4, 5, 6 og 8 komprimerer også 30 ganger, men først 10 med klikk, så 10 uten klikk til slutt 10 med klikk.

Opptak nummer:	**Lengde L:**	**Høyde H:**	**Rate:**	**Dybde:**	**Antall klikk:**	**Notat:**	**Klokkeslett:**
14	1.2m	1.8m	100/min	Normal	30 klikk i serie	Person: 1,3,7,9	
14	1.2m	1.8m	100/min	Normal	10 klikk, 10 uten klikk og 10 med klikk	Person: 2,4,5,6,8	

8.1.2 Datasett 1

Test 1

Hensikt: Teste om mikrofonen dempes/ er dynamisk.
Antall personer: 3
Oppsett: se figur 8.1.
Underlag: Teppe
Person 1: Øyvind
Person 2: Torbjørn
Person 3: Atle

1. Person 1 starter med 10 klikk, så person 2 med 10 klikk så person 3 med 10 klikk. Deretter har man 20 klikk felles. Klapp for å synkronisere lyden gjøres nær person 1.
2. Motsatt rekkefølge i forhold til opptak 01. Person 3 starter med 10 klikk, så person 2 med 10 klikk så person 1 med 10 klikk. Deretter har man 20 klikk felles. Klapp for å synkronisere lyden gjøres nær person 3.

Opptak nummer:	Avstand:	Rate:	Dybde:	Antall kompresjoner:	Notat:
01	1.2m	100/min	Normal	10 kompresjoner hver og 20 felles	Test mikrofonen Klapp med klapper utføres nær person 1
02	1.2m	100/min	Normal	10 kompresjoner hver og 20 felles	Test mikrofonen Klapp med klapper utføres nær person 3

Tabell 8.1: Opptak i Test 1

Figur 8.1: Oppsettet til Test 1 og Test 2

Figur 8.2: Disse plankene ble brukt som klapper for å synkronisere lyden i Test 1 og Test 2

Test 2

Hensikt: Teste å skille de forskjellige klikkelydene.
Antall personer: 3
Underlag: Teppe
Person 1: Øyvind
Person 2: Torbjørn
Person 3: Atle

3. Alle komprimerer i takt (stort sett) med mange glipp, altså kompresjoner uten klikk.

4. Person 1 klikker sakte, person 2 middels og person 3 raskt.

Opptak nummer:	Avstand:	Rate:	Dybde:	Antall kompresjoner:	Notat:
03	1.2m	100/min	Normal	40 kompresjoner	Skille klikkelyder
04	1.2m	100/min	Normal	40 kompresjoner	Skille klikkelyder

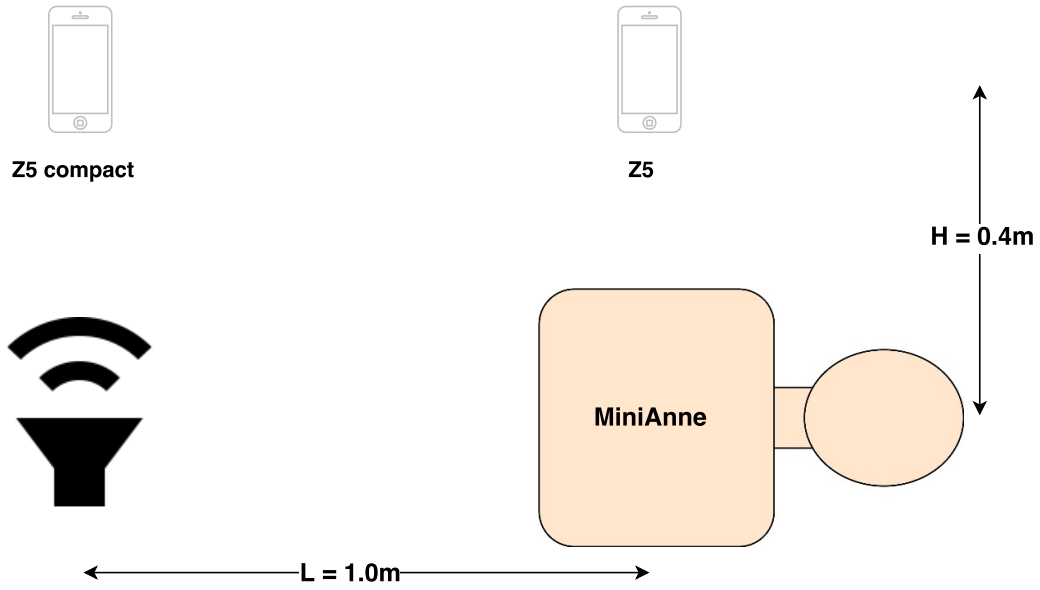
Tabell 8.2: Opptak i Test 2

8.1.3 Datasett 2

Hensikt: Gjøre opptak for å teste blind source separation.
Antall personer: 1
Antall telefoner: 2, telefon 1 = Sony Z5 compact, telefon 2 = Sony Z5
Underlag: Teppe

Z5 compact lå nærmet høytaler og Z5 lå nærmest dukke slik figur 8.3 viser. Alle lydopptak blir synkronisert med en klapper av to planker, se figur 8.2.

1. Kun kompresjoner
2. Kun musikk (*The Debutante*)
3. Kompresjoner og musikk (*The Debutante*)
4. Kompresjoner og musikk (*Ellens song*)
5. Kompresjoner og sinussignal
6. Kompresjoner og hvit støy



Figur 8.3: Testoppsett for datasett 2

8.1.4 Datasett 3

Datasett 3 har som hovedoppgave å teste forskjeller mellom dukker og dynamikken til mikrofon i telefoner.

Test 1

Hensikt: Gjøre opptak for å teste om dukkene gir forskjellige lyder og om telfonene gir forskjellige opptak.
Antall personer: 1
Oppsett: se figur 8.4.
Underlag: Laminat
Telefon 1: Blackberry Z10
Telefon 2: Sony Z5
Telefon 3: Sony Z5 compact

1. 20 kompresjoner med hver dukke 1-9.
2. 30 kompresjoner med dukke 1. Z5 compact nærmest dukke, så Z5, så BB
3. 30 kompresjoner med dukke 1. BB nærmest dukke, så Z5, så Z5 compact

Figur 8.4: Testoppsett for opptak 3, test 1

Test 2

Hensikt: Fremprovosere andre lyder som ikke oppstår under vanlig kompresjon.
Antall personer: 1
Antall telefoner: 2, Blackberry Z10 (1) og Sony Z5 compact (2)
Underlag: Teppe

Telefon 1 er blackberry og telefon 2 er Z5 compact. Z5 compact lå nærmest dukken, tilsvarende figur 8.4.

4. 60 kompresjoner med dukke 1. Komprimerer på forskjellige måter for å fremprovosere forskjellige lyder. For eksempel ble det komprimert på skrå og litt utenfor midten.

5. 60 kompresjoner med dukke 2. Komprimerer på forskjellige måter for å fremprovosere forskjellige lyder. For eksempel ble det komprimert på skrå og litt utenfor midten.

Test 3

Hensikt: Gjøre opptak for å teste klapp-påvirkning og dukkelyder.
Antall personer: 1
Antall telefoner: 2, Blackberry Z10 (1) og Sony Z5 compact (2)
Underlag: Teppe

6. Høyt klapp med klapper før komprimering. 20 kompresjoner.
7. Middels klapp med klapper før komprimering. 20 kompresjoner.
8. Lavt klapp med klapper før komprimering. 20 kompresjoner.

8.1.5 Datasett 4

Underlag: Teppe

Dette datasettet inneholder en rekke teste tatt opp med én telefon. I alle tilfeller ble templaten brukt for riktig plassering av telefon i forhold til dukke, se figur 3.1.

Opptak nummer:	Info	Hensikt:
01	40 kompresjoner med 5 kompresjoner i begynnelsen.	For å lage template med de fem første kompresjonene til testing av template matching
02	Beveger hånden opp og ned over kameraet i takt med klokka	For å finne raten basert på video
03	30 kompresjoner med mikrofonen innstilt på "Sensitivity High"	Test mikrofonen på telefonen.
04	30 kompresjoner med mikrofonen innstilt på "Sensitivity Medium"	Test mikrofonen på telefonen.
05	30 kompresjoner med mikrofonen innstilt på "Sensitivity Low"	Test mikrofonen på telefonen.
06	30 kompresjoner med mikrofonen innstilt på "Sensitivity Auto"	Test mikrofonen på telefonen.
07	Kompresjoner i 2.5 minutt med forskjellige kompresjonsrater. Fra 80 til 120 pr minutt.	Teste hvor langt det er mellom klikk ned og klikk opp, og om dette varierer med raten.
08	En del kompresjoner for å teste hvor mye støy det er på "Pergo klikk vinyl" underlag	Teste støy knyttet til underlag.

Tabell 8.3: Forskjellige tester tatt opp med kun en telefon.

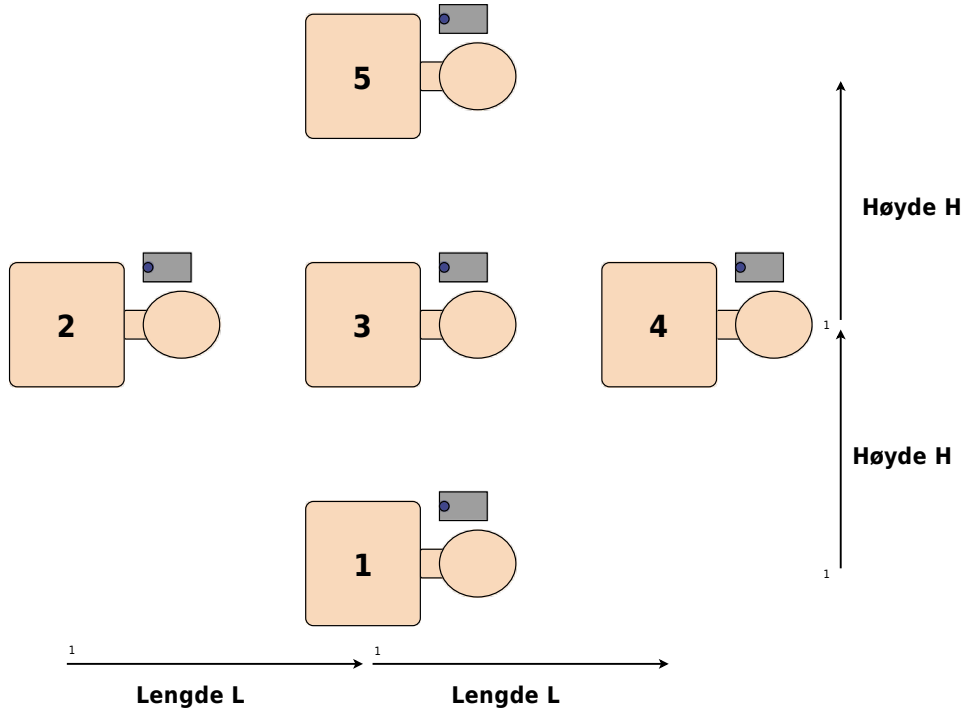
8.1.6 Datasett 5

Hensikt: Teste forskjellige avstander mellom personer for å finne ut hvilken avstand som er mulig å bruke i en fremtidig implementasjon som gir tilfredstillende deteksjonsrate.
Antall personer: 5
Oppsett: se figur 8.5 og 8.6.
Underlag: Pergo klikk vinyl
Person 1: Atle, telefon: LG G4
Person 2: Johannes, telefon: Samsung Galaxy s7 Edge
Person 3: Øyvind, telefon: Sony Xperia Z5 compact
Person 4: Åshild, telefon: LG G2
Person 5: Torbjørn, telefon: Sony Xperia Z5

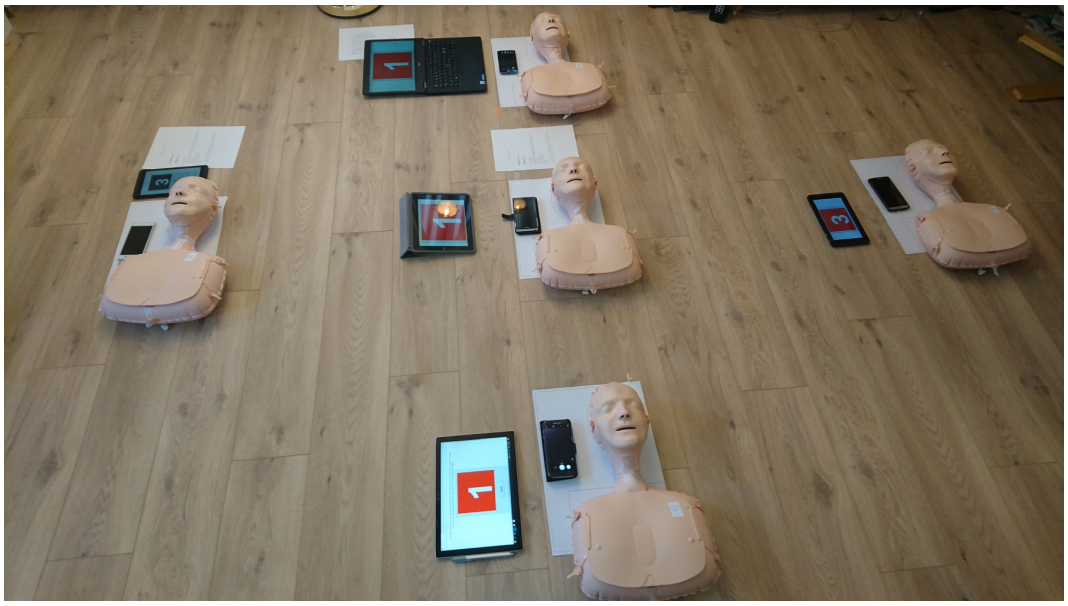
Hvert opptak synkroniseres med klapper vist i figur 8.2. Deretter gjøres 5 kompresjoner før selve opptaket starter. Hver deltaker komprimerer i sin egen hastighet fra 90 til 110 kompresjoner pr minutt: Person 1: 90/min, Person 2: 95/min, Person 3: 100/min, Person 4: 105/min, Person 5: 110/min, hhv.

Opptak nummer:	Avstand:	Rate:	Antall kompresjoner:	Notat:
01	H:1.2m L: 0.8m	90-110/min	30 kompresjoner	Teste avstand i høyde retning.
02	H:1.4m L: 0.8m	90-110/min	30 kompresjoner	Teste avstand i høyde retning.
03	H:1.6m L: 0.8m	90-110/min	30 kompresjoner	Teste avstand i høyde retning.
04	H:1.2m L: 1.0m	90-110/min	30 kompresjoner	Teste avstand i lengde retning.
05	H:1.2m L: 1.2m	90-110/min	30 kompresjoner	Teste avstand i lengde retning.
06	H:1.2m L: 1.4m	90-110/min	30 kompresjoner	Teste avstand i lengde retning.
07	H:1.4m L: 1.0m	90-110/min	30 kompresjoner	Teste avstand i begge retninger.
08	H:1.6m L: 1.2m	90-110/min	30 kompresjoner	Teste avstand i begge retninger.

Tabell 8.4: Opptak til datasett 5



Figur 8.5: Oppsettet til datasett 5 og 6



Figur 8.6: Oppsettet til datasett 5 og 6

8.1.7 Datasett 6

Hensikt: Hensikten med dette datasettet er å validere resultatene fra datasett 5. En grei høyde er valgt på 1.4 og man vil her teste mer hvor mye lengden har å si for mulighetene til å detektere klikkelyd tilhørende riktig dukke.
Antall personer: 5
Oppsett: se figur 8.5 og 8.6.
Underlag: Pergo klikk vinyl
Person 1: Atle, telefon: LG G4
Person 2: Johannes, telefon: Samsung Galaxy s7 Edge
Person 3: Øyvind, telefon: Sony Xperia Z5 compact
Person 4: Åshild, telefon: LG G2
Person 5: Torbjørn, telefon: Sony Xperia Z5

Dette datasettet er tilsvarende datasett 5 med samme personer og stort sett samme opptak. Forskjellen her er at høyden H skal være konstant på 1.4m mens lengden L varierer fra 0.8m til 1.8m. I tillegg er antall kompresjoner økt fra 30 til 60. Hvert opptak synkroniseres med klapper vist i figur 8.2. Deretter gjøres 5 kompresjoner før selve opptaket starter. Hver deltaker komprimerer i sin egen hastighet fra 90 til 110 kompresjoner pr minutt: Person 1: 90/min, Person 2: 95/min, Person 3: 100/min, Person 4: 105/min, Person 5: 110/min, hhv.

Opptak nummer:	Avstand:	Rate:	Antall kompresjoner:	Notat:
01	H:1.4m L: 0.8m	90-110/min	60 kompresjoner	Teste avstand i lengde retning.
02	H:1.4m L: 1.0m	90-110/min	60 kompresjoner	Teste avstand i lengde retning.
03	H:1.4m L: 1.2m	90-110/min	60 kompresjoner	Teste avstand i lengde retning.
04	H:1.4m L: 1.4m	90-110/min	60 kompresjoner	Teste avstand i lengde retning.
05	H:1.4m L: 1.6m	90-110/min	60 kompresjoner	Teste avstand i lengde retning.
06	H:1.4m L: 1.8m	90-110/min	60 kompresjoner	Teste avstand i lengde retning.

Tabell 8.5: Opptak til datasett 6

8.2 Kode

Vedlagt i pdf-en ligger en 7z-fil med Matlab-kode fra alle eksperimentene. Dobbelklikk på nålen for å laste ned koden,  . All koden er også listet her i dette kapittelet.

Kapittelet består av kode fra eksperiment 1-11 og tilhørende funksjoner. Funksjonenes innhold er vist i tabellen nedenfor:

- **dir_master**: Brukt for å finne sti til hvor datasettene er lagret og hvor figurer blir lagret.
- **findBestChannel**: Detekterer beste kanal. Brukes i initialisering.
- **findFiles**: Finner videofiler i en mappe og samler disse i en rxp cellestruktur hvor r er antall opptak og p er antall personer.
- **generateTemplate**: Genererer mal (template) til bruk i *template matching*. Brukes i initialisering.
- **initialisering**: Bruker første del av lydopptaket til å finne en mal (template) og flere parametre, se kapittel 4.1.1.
- **loadData**: Laster parametre som er lagret i .mat-filer.
- **makeConfusion**: Slår sammen en rxp cellestruktur av forvirringsmatriser (confusion matrix) til én forvirringsmatrise. Finner også forskjellige ytelsesmål som tAcc, PPV, NPV, Sen, Spe osv.
- **saveFigs**: En funksjon for å lagre figurer.
- **tabGen**: Genererer LaTeXtabeller direkte fra en matrise eller cellestruktur.

8.2.1 Eksperiment 1

```
% Eksperiment 1: Test av dukkelyder

clear
addpath('Funksjoner/')

[data_dir] = dir_master();  % Finner sti

folder = 'Opptak03';% Datasett 3
recordNum = 1;      % Opptak nummer
personNum = 1:3;    % Personer

delayFirst = 48000;   % Starten av lydopptaktet er 1sekund etter klapping

filtA = designfilt('bandpassiir', 'StopbandFrequency1', 1000,...
    'PassbandFrequency1', 2000, 'PassbandFrequency2', 15000,...
    'StopbandFrequency2', 16000, 'StopbandAttenuation1', 40,...
    'PassbandRipple', 1, 'StopbandAttenuation2', 40, 'SampleRate',...
    48000, 'DesignMethod', 'ellip');



% Laster cutpos = synkroniseringstidspunkt og annoterte kanaler
Names = {'cutpos','trueCanals'};
[cutpos,trueCanals] = loadData(folder,Names);



files = dir(strcat(data_dir,folder,'\'));
[fileNames,R,P,N] = findFiles(files);

for p = personNum

    filename = fileNames{recordNum,p};
    filename = strcat(data_dir,folder,'\',filename);
    [y,Fs] = audioread(filename);   % Laster fil

```

```
37      if ¬(Fs == 48000)   % Resampler om ndvendig
38          y = resample(y,48000,Fs);
39          Fs = 48000;
40      end
41      % Synkroniseringstidspunkt og valg av kanal:
42      y = y(cutpos(recordNum,p)+delayFirst:end,trueCanals(recordNum,p))';
43
44      data{p} = y;
45  end
46  %% Forskjeller fra dukke til dukke
47  figure(1),clf
48
49  subplot(3,1,1)
50  plot(linspace(0,length(data{3}(1e6:1.5e6))/Fs,...
51      length(data{3}(1e6:1.5e6))),data{3}(1e6:1.5e6))
52  title(['Forskjeller fra dukke til dukke: Datasett 3, opptak '...
53      num2str(recordNum) ', telefon ' num2str(3) ', dukke 2'],...
54      'FontSize',16,'FontWeight','bold')
55  legend('x(n)')
56  xlabel('Tid[s]')
57  ylabel('Amplitude')
58  axis([0 10 0 1.1])
59
60
61  subplot(3,1,2)
62  plot(linspace(0,length(data{3}(1.8e6:2.3e6))/Fs,...
63      length(data{3}(1.8e6:2.3e6))),data{3}(1.8e6:2.3e6))
64  title(['Datasett 3, opptak ' num2str(recordNum) ', telefon ' num2str(3)...
65      ', dukke 3'],'FontSize',16,'FontWeight','bold')
66  legend('x(n)')
67  xlabel('Tid[s]')
68  ylabel('Amplitude')
69  axis([0 10 0 1.1])
70
71  subplot(3,1,3)
72  plot(linspace(0,length(data{3}(5.45e6:5.95e6))/Fs,...
73      length(data{3}(5.45e6:5.95e6))),data{3}(5.45e6:5.95e6))
74  title(['Datasett 3, opptak ' num2str(recordNum) ', telefon '...
75      num2str(3) ', dukke 7'],'FontSize',16,'FontWeight','bold')
76  legend('x(n)')
```

```
77 xlabel('Tid[s]')
78 ylabel('Amplitude')
79 axis([0 10 0 1.1])
80 set(gcf, 'Position', [2575, 200, 1000, 600]);  % Set size
81 % saveFigs('forskjellDukke',gcf,2,1,[2575, 400, 1000, 600])
82
83
84
85
86 %% Kompresjoner p  forskjellig vis
87
88 recordNum = 4;
89 personNum = 1:2;
90
91 figure(2),clf
92 for p = personNum
93
94     filename = fileNames{recordNum,p};
95     filename = strcat(data_dir,folder,'\',filename);
96     [y,Fs] = audioread(filename);   % Laster fil
97
98     if ¬(Fs == 48000)   % Resampler om ndvendig
99         y = resample(y,48000,Fs);
100        Fs = 48000;
101    end
102
103    % Synkroniseringstidspunkt og valg av kanal:
104    y = y(cutpos(recordNum,p)+delayFirst:end,trueCanals(recordNum,p));
105
106    y = y(1:2e6);
107    subplot(2,1,p)
108    plot(linspace(0,length(y)/Fs,length(y)),y)
109    axis([0 41 0 1.1])
110    if p == 1
111        title(['Datasett 3, opptak ' num2str(recordNum) ', telefon '...
112            num2str(p)],'FontSize',16,'FontWeight','bold')
113        legend('x(n)')
114        xlabel('Tid[s]')
115        ylabel('Amplitude')
116    else
```

```
117         title(['Datasett 3, opptak: ' num2str(recordNum) ', telefon '...
118             num2str(p)],'FontSize',16,'FontWeight','bold')
119         legend('x(n)')
120         xlabel('Tid[s]')
121         ylabel('Amplitude')
122     end
123 end
124 set(gcf, 'Position', [2575, 600, 1000, 600]);  % Set size
125 % saveFigs('spesielleDukkelyder',gcf,2,1,[2575, 400, 1000, 600])
```

8.2.2 Eksperiment 2

```
% Eksperiment 2: Test av telefoner

clear
addpath('Funksjoner/')

[data_dir,img_dir] = dir_master();

folder = 'Opptak03';% Datasett 3
recordNum = 1;      % Opptak 1
personNum = 1:3;    % Person 1, 2 og 3

delayFirst = 48000; % Starten av lydopptaktet er 1 sekund etter klapping


% Genererer filter
filtA = designfilt('bandpassiir', 'StopbandFrequency1', 1000,...
    'PassbandFrequency1', 2000, 'PassbandFrequency2', 15000,...
    'StopbandFrequency2', 16000, 'StopbandAttenuation1', 40,...
    'PassbandRipple', 1, 'StopbandAttenuation2', 40, 'SampleRate',...
    48000, 'DesignMethod', 'ellip');



% Laster cutpos = synkroniseringstidspunkt og annoterte kanaler
Names = {'cutpos','trueCanals'};
[cutpos,trueCanals] = loadData(folder,Names);


% Finner filer i mappe
files = dir(strcat(data_dir,folder,'\'));
[fileNames,R,P,N] = findFiles(files);


%% Genererer figur
figure(1),clf

```

```
for p = personNum
    filename = fileNames{recordNum,p};
    filename = strcat(data_dir,folder,'\',filename);
    [y,Fs] = audioread(filename);

    if ¬(Fs == 48000)   % Resampler om ndvendig
        y = resample(y,48000,Fs);
        Fs = 48000;
    end
    y = y(cutpos(recordNum,p)+delayFirst:end,trueCanals(recordNum,p))';


    dataForPlot = y(1e6:1.5e6); % Tidsintervall brukt i plot
    subplot(3,1,p)
   plot(linspace(0,length(dataForPlot)/Fs,length(dataForPlot)),dataForPlot)
    if p == 1
        title(['Forskjell fra telefon til telefon, dukke 2: Datasett 3,'...
            ' opptak ' num2str(recordNum) ', telefon ' num2str(p)],...
            'FontSize',16,'FontWeight','bold')
    else
        title(['Telefon ' num2str(p)],'FontSize',16,'FontWeight','bold')

    end
    legend('x(n)')
    xlabel('Tid[s]')
    ylabel('Amplitude')
    axis([0 10 0 1.1])
end
set(gcf, 'Position', [2575, 400, 1000, 600]);  % Set size
% saveFigs('forskjellTelefon',gcf,2,1,[2575, 400, 1000, 600])
```

8.2.3 Eksperiment 3

```
% Eksperiment 3: Kanaldeteksjon

clear,clc
[data_dir,¬] = dir_master();

folder = 'Opptak05';% Datasett
r = 6;              % Opptak
p = 2;              % Person

delayFirst = 48000;% Starten av lydopptaktet er 1 sekund etter klapping


% Laster cutpos = synkroniseringstidspunkt og annoterte kanaler
[cutpos,trueCanals] = loadData(folder,{'cutpos','trueCanals'});

% Finner filer i mappe
files = dir(strcat(data_dir,folder,'\'));
[fileNames,R,P,N] = findFiles(files);


filename = fileNames{r,p};
filename = strcat(data_dir,folder,'\',filename);
[y,Fs] = audioread(filename);

y = y(cutpos(r,p)+delayFirst:end,:);

if ¬(Fs == 48000)   % Resampler om ndvendig
    y = resample(y,48000,Fs);
    Fs = 48000;
end
y1 = y(:,1);
y2 = y(:,2);


%% Generer plot

```

```
figure(1),clf
subplot(2,1,1)
plot(linspace(0,length(y1)/Fs,length(y1)),y1)
hold on
plot([10 10], [-1.1 1.1])
axis([0 34 -1.1 1.1])
title(['Kanal 1: Datasett 5, opptak ' num2str(r) ' og person '...
    num2str(p)],'FontSize',16,'FontWeight','bold')
legend({'x(n)','Grense for initialisering'},'FontSize',10)
xlabel('Tid[s]')
ylabel('Amplitude')

subplot(2,1,2)
plot(linspace(0,length(y2)/Fs,length(y2)),y2)
hold on
plot([10 10], [-1.1 1.1])
axis([0 34 -1.1 1.1])
title('Kanal 2:','FontSize',16,'FontWeight','bold')
legend({'x(n)','Grense for initialisering'},'FontSize',10)
xlabel('Tid[s]')
ylabel('Amplitude')

set(gcf, 'Position', [2575, 400, 1000, 700]);  % Set size
%         saveFigs('kanalForskjell',gcf,2,0,[2575, 400, 1000, 700])


%% Generer tabeller:

% Kjrer gjennom alle datasett og alle opptak
folders = {'Opptak05','Opptak06','Laerdal'};
% folders = {'Opptak06'};

for i = 1:length(folders)
    if strcmp(folders{i},'Laerdal')
        recordNums{i} = 1:14;
        personNums{i} = 1:8;
    elseif strcmp(folders{i},'Opptak05')
        recordNums{i} = 1:8;
        personNums{i} = 1:5;
    elseif strcmp(folders{i},'Opptak06')
```

```
77          recordNums{i} = 1:6;
78          personNums{i} = 1:5;
79      end
80  end
81
82  % Genererer filter
83  filtA = designfilt('bandpassiir', 'StopbandFrequency1', 1000,...
84      'PassbandFrequency1', 2000, 'PassbandFrequency2', 15000,...
85      'StopbandFrequency2', 16000, 'StopbandAttenuation1', 40,...
86      'PassbandRipple', 1, 'StopbandAttenuation2', 40, 'SampleRate',...
87      48000, 'DesignMethod', 'ellip');
88
89
90  for f = 1:length(folders)
91      folder = folders{f};
92      files = dir(strcat(data_dir,folder,'\'));
93      [fileNames,R,P,N] = findFiles(files);
94
95      [cutpos,trueCanals] = loadData(folder,{'cutpos','trueCanals'});
96
97      detectedChannel = [];
98      for r = recordNums{f}
99          for p = personNums{f}
100             filename = fileNames{r,p};
101             [y,Fs] = audioread(strcat(data_dir,folder,'\',filename));
102             delayFirst = Fs;
103             y = y(cutpos(r,p)+delayFirst:end,:);
104
105             if ¬(Fs == 48000)   % Resampler
106                 y = resample(y,48000,Fs);
107                 Fs = 48000;
108             end
109
110             [¬,bestChannel,¬,¬,¬,¬] = initialisering(y,filtA,filename,...
111                 folder,trueCanals,0); % Detekterer beste kanal
112
113             % Lagrer beste kanal i en matrise
114             detectedChannel(r,p) = bestChannel;
115
116         end
```

```
117     end
118
119     % Fjerner alle iPhone opptak hvor det kun er en kanal
120     if strcmp(folder,'Laerdal')
121         detectedChannel(:,[1 3 6]) = [];
122         trueCanals(:,[1 3 6]) = [];
123     end
124     % sammenligner detekterte og annoterte kanaler og lagrer det i en
125     % cellestruktur result
126     result{f} = detectedChannel==trueCanals(1:size(detectedChannel,1),...
127         1:size(detectedChannel,2));
128
129 end
130
131
132 %% Lager tabell fra alle datasett
133 dec = 4;    % Antall gjeldende siffer som skal vises i tabell
134 tabell{1,1} = size(result{1},1)*size(result{1},2); % Antall opptak
135 tabell{1,2} = size(result{2},1)*size(result{2},2); % Antall opptak
136 tabell{1,3} = size(result{3},1)*size(result{3},2); % Antall opptak
137 tabell{2,1} = sum(sum(result{1})); % Antall riktig detekterte kanaler
138 tabell{2,2} = sum(sum(result{2})); % Antall riktig detekterte kanaler
139 tabell{2,3} = sum(sum(result{3})); % Antall riktig detekterte kanaler
140
141 % Andel riktig detekterte kanaler
142 tabell{3,1} = [num2str(tabell{2,1}/tabell{1,1}*100,dec) '\%'];
143 tabell{3,2} = [num2str(tabell{2,2}/tabell{1,2}*100,dec) '\%'];
144 tabell{3,3} = [num2str(tabell{2,3}/tabell{1,3}*100,dec) '\%'];
145
146
147 leftTitle = {'Antall opptak:','Antall riktig detekterte kanaler:',...
148     'Prosent riktig detekterte kanaler:'};
149 topTitle = {'Resultater:','Datasett 5','Datasett 6','Datasett Laerdal'};
150 label = 'tab:kanaldeteksjon';
151 caption = ['Tabellen viser hvor mange opptak i hvert datasett hvor'...
152     ' riktig kanal ble detektert.'];
153 tabGen(tabell,topTitle,leftTitle,dec,caption,label,6);
154
155
156
```

```
157
158
159 %% Tabell Laerdal
160 leftTitle = {'Opptak 1','Opptak 2','Opptak 3','Opptak 4','Opptak 5',...
161     'Opptak 6','Opptak 7','Opptak 8','Opptak 9','Opptak 10',...
162     'Opptak 11','Opptak 12','Opptak 13','Opptak 14'};
163 topTitle = {'Opptak/Person nr.','2','4','5','7','8'};
164 label = 'tab:kanaldeteksjon2';
165 caption = ['Resultat av kanaldeteksjon i datasett Laerdal.'...
166     ' 1 betyr at riktig kanal ble detektert og 0 betyr at feil kanal '...
167     'ble detektert. Person 1, 3 og 6 brukte iPhone som kun har en'...
168     'kanal og er derfor ikke tatt med i resultatene her.'];
169 tabGen(result{3},topTitle,leftTitle,dec,caption,label,6);
```

8.2.4 Eksperiment 4

```
% Eksperiment 4: Valg av forskjellige parametre

clear,clc
folder = 'Opptak05';
warning off

plotFigures = 0;
folders = {'Opptak05','Opptak06'};
recordNums = {1:8,1:6};
personNums = {1:5,1:5};

% trueRightClick = annotert klikk opp
% trueLeftClick = annotert klikk ned
% maxValue = "maksVerdi"
% y_upper = x_env(n), envelopen til x(n)
Names = {'trueRightClick','trueLeftClick','maxValue','y_upper'};

% result: forhold mellom amplitude p  annotert klikk og maksVerdi
result = [];

% result2: avstand mellom klikk ned og klikk opp p  datasett 5 og 6
result2 = [];

for f = 1:length(folders)
    folder = folders{f};
    recordNum = recordNums{f};
    personNum = personNums{f};

    % Laster parametre
    [annotertKlikkOpp,annotertKlikkNed,maksVerdi,x_env]...
        = loadData(folder,Names);

    for r = recordNum
        for p = personNum
            right = annotertKlikkOpp{r,p};
            left = annotertKlikkNed{r,p};
```

```
37
38
39              % forhold mellom amplitude p annotert klikk og maksVerdi:
40              allClick = [left right];
41              [B,I] = sort(allClick);
42              allClick = allClick(I);
43              result=[result x_env{r,p}(allClick)./maksVerdi{r,p}(allClick)];
44
45
46
47              % avstand mellom klikk ned og klikk opp p datasett 5 og 6:
48              [¬,Ir] = sort(right);
49              [¬,Il] = sort(left);
50              rightSorted = right(Ir);
51              leftSorted = left(Il);
52              dist = rightSorted-leftSorted;
53              result2 = [result2 (dist)];
54
55
56          end
57      end
58
59  end
60  %% Forhold mellom amplitude p annoterte klikk og maksVerdi
61  figure(1),clf
62  plot([0.3 0.3],[0,280],'r')
63  hold on
64  histogram(result,500)
65  title('Forhold mellom amplitude p annoterte klikk og maksVerdi',...
66      'FontSize',16)
67  legend({'Terskelfaktor'},'FontSize',12)
68  xlabel('Amplitude / maxValue')
69  ylabel('Antall')
70  axis([0 2 0 280])
71  set(gcf, 'Position', [2575, 400, 1000, 600]);  % Set size
72  % saveFigs('terskelfaktor',gcf,2,1,[2575, 400, 1000, 600])
73
74
75  %% Avstand mellom klikk ned og klikk opp p datasett 5 og 6
76  figure(2),clf
```

```
77 plot([3000 3000],[0 140],'r')
78 hold on
79 plot([12000 12000],[0 140],'g')
80 histogram(result2,100)
81 title('Avstand mellom klikk ned og klikk opp p  datasett 5 og 6',...
82     'FontSize',16)
83 xlabel('Avstand i sampler')
84 ylabel('Antall kompresjoner')
85 legend({'enkelKlikkDistanseMin','enkelKlikkDistanseMaks'},'FontSize',12)
86 axis([2000 14000 0 140])
87 set(gcf, 'Position', [2575, 400, 1000, 600]);  % Set size
88 % saveFigs('AvstandMellomKlikk',gcf,2,1,[2575, 400, 1000, 600])
```

8.2.5 Eksperiment 5

```
% Eksperiment 5: Valg av båndpassfilter

clear,clc

% Laster data, yuse som er alle opptak i datasett 5 satt sammen til ett
% lydsignal:
load('C:\Users\oyvin\Dropbox\Masteroppgave\Matlab\Opptak05All.mat')
[data_dir] = dir_master();

% Genererer filter
filtA = designfilt('bandpassiir', 'StopbandFrequency1', 1000,...
    'PassbandFrequency1', 2000, 'PassbandFrequency2', 15000,...
    'StopbandFrequency2', 16000, 'StopbandAttenuation1', 40,...
    'PassbandRipple', 1, 'StopbandAttenuation2', 40, 'SampleRate',...
    48000, 'DesignMethod', 'ellip');

Fs = 48000; % Samplerate

yBP = filter(filtA,yuse);   % Båndpassfiltrerer

signal1_fft = abs(fft(yBP,Fs)); % Finner frekvensspekter
signal_fft = abs(fft(yuse,Fs)); % Finner frekvensspekter


%% Plot av båndpassfilter
figure(1),clf
subplot(2,1,1)
plot(signal_fft(1:Fs/2))
title('Frekvensspekter av hele datasett 5','FontSize',16,...
    'FontWeight','bold')
axis([0 Fs/2 0 15])

subplot(2,1,2)
plot(signal1_fft(1:Fs/2))
title('Frekvensspekter av datasett 5 filtrert med båndpassfilter',...
    'FontSize',16,'FontWeight','bold')
```

```
37 axis([0 Fs/2 0 4])
38
39 set(gcf, 'Position', [2575, 400, 1000, 500]);  % Set size
40 %     saveFigs('FFTDatasett5',gcf,2,1,[2575, 400, 1000, 500])
41
42 %% Filtrering av talesty
43
44 % Laster data
45 folder = 'Laerdal';     % Datasett Laerdal
46 [trueCanals] = loadData(folder,{'trueCanals'});% Laster annoterte kanaler
47 r = 6;  % Opptak nummer
48 p = 8;  % Person nummer
49 files = dir(strcat(data_dir,folder,'\'));   % Finner filer i mappen
50 [fileNames] = findFiles(files);   % Finner aktuelle filer
51
52
53 filename = fileNames{r,p};
54 filename = strcat(data_dir,folder,'\',filename);
55 [y,Fs] = audioread(filename);   % Laster fil
56 y = y(:,trueCanals(r,p))';  % Velger riktig kanal
57 start = 43.75*Fs;           % Start posisjon
58 stopp = start+1.5*Fs;       % Stopp posisjon
59 lengde = (stopp-start)/Fs;  % Lengde p  opptak
60 y = y(1,start:stopp);
61
62
63
64
65 yfilter = filter(filtA,y);  % Filtrerer lydsignalet
66
67 % Finner toppunkt til klikk i ufiltrert lydopptak
68 [pks1,locs1] = findpeaks(y,'MinPeakHeight',0.5);
69 behold1 = [2 4 6 7 8 10];
70 locs1 = locs1(behold1);
71 pks1 = pks1(behold1);
72
73 % Finner toppunkt til klikk i filtrert lydopptak
74 [pks2,locs2] = findpeaks(yfilter,'MinPeakHeight',0.4);
75 behold2 = [1 2 4 5 8 10];
76 locs2 = locs2(behold2);
```

```
77 pks2 = pks2(behold2);
78
79
80 % Plotter ufiltrert signal
81 figure(2),clf
82 plot(linspace(0,length(y)/Fs,length(y)),y)
83 hold on
84 scatter(locs1/Fs,pks1)
85 axis([0, lengde, -1 0.8])
86 title(['Originalt lydsignal x(n)'],'FontSize',16,'FontWeight','bold')
87 legend({'x(n)','Klikk'},'FontSize',12)
88 xlabel('Tid[s]')
89 ylabel('Amplitude')
90 set(gcf, 'Position', [2575, 600, 1000, 800]);  % Set size
91 %     saveFigs('yufiltrert',gcf,2,1,[2575, 600, 1000, 800])
92
93
94 % Plotter filtrert signal
95 figure(3),clf
96 plot(linspace(0,length(yfilter)/Fs,length(yfilter)),yfilter)
97 hold on
98 scatter(locs2/Fs,pks2)
99 axis([0, lengde, -1 0.8])
100
101 title(['Filtrert lydsignal, x_{bp}(n)'],'FontSize',16,'FontWeight','bold')
102 legend({'x_{bp}(n)','Klikk'},'FontSize',12)
103 xlabel('Tid[s]')
104 ylabel('Amplitude')
105 set(gcf, 'Position', [2575, 200, 1000, 800]);  % Set size
106 %     saveFigs('yfiltrert',gcf,2,1,[2575, 200, 1000, 800])
107
108
109
110 %% Filtrerer st y
111
112 % Laster inn data
113 folder = 'Opptak05';
114 Names = {'trueCanals','y_upper'};
115 [trueCanals,y_upper] = loadData(folder,Names);
116 r = 4;
```

```
117 p = 4;
118
119
120 files = dir(strcat(data_dir,folder,'\'));   % Finner filer i mappen
121 [fileNames,R,P,N] = findFiles(files);   % Finner aktuelle filer
122
123 filename = fileNames{r,p};
124 filename = strcat(data_dir,folder,'\',filename);
125
126 [y,Fs] = audioread(filename);   % Laster fil
127
128 y = y(:,trueCanals(r,p))';  % Bruker kun en kanal
129
130 start = 20*Fs;      % Start p  lydopptaket
131 stopp = start+3*Fs; % Stopp p  lydopptaket
132
133
134 lengde = (stopp-start)/Fs; % Lengde p  lydopptaket
135 y = y(1,start:stopp);
136
137 yfilter = filter(filtA,y);  % Filtrer
138
139 % Finner toppunkt til klikk i ufiltrert lydopptak
140 [pks1,locs1] = findpeaks(y,'MinPeakHeight',0.75,'MinPeakDistance',2000);
141 behold1 = [1 3 4 5 6 8 10 11 13 14 18 19];
142 locs1 = locs1(behold1);
143 pks1 = pks1(behold1);
144
145 % Finner toppunkt til klikk i filtrert lydopptak
146 [pks2,locs2] = findpeaks(yfilter,'MinPeakHeight',0.65,...
147     'MinPeakDistance',5000);
148 behold2 = [1:8 10 11 13 14];
149 locs2 = locs2(behold2);
150 pks2 = pks2(behold2);
151
152
153
154 % Plotter ufiltrert signal
155 figure(4),clf
156 plot(linspace(0,length(y)/Fs,length(y)),y)
```

```
157 hold on
158 scatter(locs1/Fs,pks1)
159 axis([0, lengde, -1.2 1.1])
160 title(['Datasett 5, opptak 4, person 4. Originalt lydsignal x(n)'],...
161     'FontSize',16,'FontWeight','bold')
162 legend({'x(n)','Klikk'},'FontSize',12)
163 xlabel('Tid[s]')
164 ylabel('Amplitude')
165 set(gcf, 'Position', [2575, 500, 1000, 800]);  % Set size
166 %     saveFigs('yufiltrert2',gcf,2,1,[2575, 500, 1000, 800])
167
168 % Plotter filtrert signal
169 figure(5),clf
170 plot(linspace(0,length(yfilter)/Fs,length(yfilter)),yfilter)
171 hold on
172 scatter(locs2/Fs,pks2)
173 axis([0, lengde, -1.2 1.1])
174 title(['Datasett 5, opptak 4, person 4. Filtrert lydsignal, x_{bp}(n)'],...
175     'FontSize',16,'FontWeight','bold')
176 legend({'x_{bp}(n)','Klikk'},'FontSize',12)
177 xlabel('Tid[s]')
178 ylabel('Amplitude')
179 set(gcf, 'Position', [2575, 200, 1000, 800]);  % Set size
180 %     saveFigs('yfiltrert2',gcf,2,1,[2575, 200, 1000, 800])
```

8.2.6 Eksperiment 6

```
% Eksperiment 6: Valg av avstand mellom gaussfunksjoner i mal

clear,clc

recordNum = 1;
personNum = 3;
plotFigures = 1;
folder = 'Opptak05';
r = 6;
p = 5;



Fs = 48000;
Names = {'trueRightClick','trueLeftClick','maxValue','y_upper'};
[trueRightClick,trueLeftClick,maxValue,y_upper] = loadData(folder,Names);

threshValue = 0.3*maxValue{r,p};

y = y_upper{r,p};
dist = [5000 5700 6400];

D = size(dist,2);
plotNum = D+1;



right = trueRightClick{r,p};
left = trueLeftClick{r,p};
[~,Ir] = sort(right);
[~,Il] = sort(left);
rightSorted = right(Ir);
leftSorted = left(Il);
dist = rightSorted-leftSorted;

```

```
37
38 distMean = round(mean(dist));
39 distUse = [distMean-700 distMean distMean+700];
40
41
42
43
44 figure(1),clf
45 subplot(plotNum,1,1)
46 plot(linspace(0,length(y)/Fs,length(y)),y)
47 title(['Template Matching: Datasett 5, opptak: ' num2str(r) ', person: '...
48     num2str(p)],'FontSize',16,'FontWeight','bold')
49 legend({'x_{env}(n)'},'FontSize',12)
50 axis([0 23 -0.1 1.1])
51 xlabel('Tid[s]')
52 ylabel('Amplitude')
53
54
55 for d = 1:D
56     [ template ] = generateTemplate(distUse(d),[1,1],2000,0);
57
58     xconv = conv(abs(y),template,'same');       % Konvolusjon
59
60     subplot(plotNum,1,d+1)
61     plot(linspace(0,length(xconv)/Fs,length(xconv)),xconv)
62     if d == 2
63         title(['Avstand mellom gaussfunksjoner i mal: '...
64             num2str(distUse(d)) ' sampler (Gjennomsnitt)'],...
65             'FontSize',16,'FontWeight','bold')
66     else
67         title(['Avstand mellom gaussfunksjoner i mal: '...
68             num2str(distUse(d)) ' sampler'],'FontSize',16,...
69             'FontWeight','bold')
70     end
71     legend({'x_{conv}(n)'},'FontSize',12)
72     xlabel('Tid[s]')
73     ylabel('Amplitude')
74     axis([0 23 0 550])
75 end
76
```

```
77 set(gcf, 'Position', [2575, 400, 1000, 800]);  % Set size
78 %     saveFigs('TemplateMatcingResult',gcf,2,1,[2575, 400, 1000, 800])
```

8.2.7 Eksperiment 7

```
% Eksperiment 7: Valg av bredde p a gaussfunksjoner i mal

clear,clc
folder = 'Opptak05';% Datasett 3
r = 6;              % Opptak 1
p = 5;              % Person 3



Fs = 48000; % Starten av lydopptaktet er 1 sekund etter klapping

% trueRightClick = annotert klikk opp
% trueLeftClick = annotert klikk ned
% maxValue = "maksVerdi"
% y_upper = x_env(n), envelopen til x(n)
Names = {'trueRightClick','trueLeftClick','maxValue','y_upper'};


[trueRightClick,trueLeftClick,maxValue,y_upper] = loadData(folder,Names);

x_conv = y_upper{r,p};


% Sorterer annoterte klikk ned og annoterte klikk opp i stigende rekkeflge
[~,Ir] = sort(trueRightClick{r,p});
[~,Il] = sort(trueLeftClick{r,p});
rightSorted = right(Ir);
leftSorted = left(Il);

% Avstand mellom klikk ned og klikk opp
dist = rightSorted-leftSorted;

% Gjennomsnittlig avstand mellom klikk ned og klikk opp
distMean = round(mean(dist));


```

```
37
38 %% Plot avstand mellom klikk ned og klikk opp
39 figure(1),clf
40 plot(dist)
41 title(['Avstand mellom klikk ned og klikk opp, gjennomsnitt = '...
42     num2str(distMean) ' sampler'],'FontSize',16,'FontWeight','bold')
43 xlabel('Kompresjon nummer')
44 ylabel('Antall sampler avstand mellom klikk ned og klikk opp')
45 set(gcf, 'Position', [2575, 400, 1000, 600]);  % Set size
46 %     saveFigs('TemplateMatcingDistance',gcf,2,1,[2575, 400, 1000, 600])
47
48
49
50 %% TemplateBredde
51 bredde = [400 1000 2000];   % Bredde p  malen
52 D = length(bredde); % Antall bredder
53 plotNum = D+1;      % Antall subplot
54
55 figure(2),clf
56 subplot(plotNum,1,1)
57 plot(linspace(0,length(x_conv)/Fs,length(x_conv)),x_conv)
58 title(['Template Matching: Datasett 5, opptak: ' num2str(r) ', person: '...
59     num2str(p)],'FontSize',16,'FontWeight','bold')
60 xlabel('Tid[s]')
61 ylabel('Amplitude')
62 legend({'x_{env}(n)'},'FontSize',12)
63 axis([0 23 -0.1 1.1])
64
65 akseHoyde = [300 500 600 600]; % Aksehyde
66 for d = 1:D
67     % Genererer mal (template)
68     [ template ] = generateTemplate(distMean,[1,1],bredde(d),0);
69
70     % Konvolverer
71     xconv = conv(abs(x_conv),template,'same');  % Konvolusjon
72
73     subplot(plotNum,1,d+1)
74     plot(linspace(0,length(xconv)/Fs,length(xconv)),xconv)
75     title(['Bredde gaussfunksjoner i mal: ' num2str(bredde(d))...
76         ' sampler'],'FontSize',16,'FontWeight','bold')
```

```
77      xlabel('Tid[s]')
78      ylabel('Amplitude')
79      legend({'x_{conv}(n)'},'FontSize',12)
80      axis([0 23 0 akseHoyde(d)])
81  end
82
83  set(gcf, 'Position', [2575, 400, 1000, 800]);  % Set size
84  %      saveFigs('TemplateBredde',gcf,2,1,[2575, 400, 1000, 800])
```

8.2.8 Eksperiment 8

```
% Eksperiment 8: Blind Source separation


clear,clc
addpath('Funksjoner/')
[data_dir,img_dir] = dir_master();

folder = 'Opptak02';


files = dir(strcat(data_dir,folder,'\'));
[fileNames,R,P,N] = findFiles(files);


% Forhndsdefinerer en X med miksede signaler som i hvert fall er stor nok
X = zeros(1,10000000);


% Samplerate, resampler til denne frekvensen som er standard p  Android
% telefoner. iPhone kjrer  p  44.1kHz og blir derfor resamplet til 48kHz.
FsTarget = 48000;
load(strcat(data_dir,folder,'\','cutpos.mat'))%Kuttposisjon, synkronisering



savefigures = 0;    % [1] Lagre figurer [0] ikke lagre
eksperiment = 'b';  % Eksperiment ['a'] eller ['b']



%% Eksperiment 8a Kildesignal mikset i mikrofon
if  strcmp(eksperiment,'a')
    % Setter sammen miksede signal i en matrise X:

    recordNum = 3;
```

```
37      startPos = 20.5;
38      lengde = 4;
39      personer = [1 2];
40      opptak = [recordNum recordNum];
41
42      for i = P'  % For hver person
43          filename = fileNames{recordNum,i}
44          [x,Fs] = audioread(strcat(data_dir,folder,'\',filename));
45          x = x';
46
47          x = x(1,cutpos(recordNum,i)+1:end);
48
49          if ¬(Fs == FsTarget)
50              x = resample(x,FsTarget,Fs); % Resampler til 48kHz
51          end
52
53          % Legger sammen hvert av de miksede signalene i en
54          N = min(size(x,2),size(X,2));
55          X(:,N+1:end) = [];
56          x(N+1:end) = [];
57
58          X(end+1,1:N) = x'; % Miksede lydsignaler x settes i matrise X
59
60      end
61      X(1,:) = [];
62      X = X(:,Fs*startPos:Fs*(startPos+lengde)-1);
63
64
65
66
67      %% Eksperiment 8b Digitalt mikset kildesignal
68  elseif strcmp(eksperiment,'b')
69      % Setter sammen miksede signal i en matrise X:
70
71      recordNum = 3;  % Opptak nummer
72      startPos = 14;  % Start posisjon i sekund
73      lengde = 4;     % Lengde p  signalet
74      personer = [2 1];
75      opptak = [1 2];
76
```

```
77      % Laster miksede lydopptak
78      [y1,Fs_(1)] = audioread(strcat(data_dir,folder,'\',...
79          fileNames{opptak(1),personer(1)}));
80      [y2,Fs_(2)] = audioread(strcat(data_dir,folder,'\',...
81          fileNames{opptak(2),personer(2)}));
82
83      % Resampler dersom de er forskjellig
84      Fs = max(Fs_);
85      if Fs_(1)== Fs_(2)
86      elseif Fs_(1)<Fs_(2)
87          y1 = resample(y1,Fs_(2),Fs_(1));
88      elseif Fs_(1)>Fs_(2)
89          y2 = resample(y2,Fs_(1),Fs_(2));
90      end
91      M = min(size(y1,1),size(y2,1));
92      x = [y1(1:M,1),y2(1:M,1)];
93
94      x = x(Fs*startPos:Fs*(startPos+lengde)-1,:);
95      sig = x';
96
97      %     Aorig=rand(size(sig,1));          % Miksematrise finnes tilfeldig
98      Aorig = [0.4218,0.7922;0.9157,0.9595];  % Her brukes den samme om igjen
99      X=(Aorig*sig);            % Lydsignalet mikses med miksematrisen
100 end
101
102
103
104 % Utfrer BSS med FastICA algoritmen:
105 % U = Estimat av S (kildesignaler)
106 % A = Estimat av miksematrisen A
107 % W = Separasjonsmatrise W
108 [U,A,W] = fastica(X, 'numOfIC', 'symm', 'g', 'tanh');
109
110
111
112
113 %% Generer figurer
114 % Originalt lydopptak
115 figure(1),clf
116 n = size(sig,1);
```

```
117 if ¬exist('sig','var')
118     sig = X;
119 end
120 for i = 1:n
121     subplot(n,1,i)
122     plot(linspace(0,length(sig(i,:))/Fs,length(sig(i,:))),sig(i,:))
123     if i==1
124         title(['Originale lydopptak'],'FontSize',16,'FontWeight','bold')
125     end
126 
127     legend({['s_' num2str(i) '(n), opptak ' num2str(opptak(i))...
128         ', person ' num2str(personer(i))]},'FontSize',10);
129     xlabel('Tid[s]')
130     ylabel('Amplitude')
131     set(gcf, 'Position', [2575, 400, 800, 600]);  % Set size
132 end
133 if savefigures  % Lagrer figuren
134     saveFigures('OriginaleSignaler',eksperiment)
135 end
136 
137 % Mikset lydopptak
138 figure(2),clf
139 n = size(X,1);
140 for i = 1:n
141     subplot(n,1,i)
142     plot(linspace(0,length(X(i,:))/Fs,length(X(i,:))),X(i,:))
143     if i==1
144         title(['Miksede lydopptak'],'FontSize',16,'FontWeight','bold')
145     end
146     legend({['x_' num2str(i) '(n), opptak ' num2str(opptak(i))...
147         ', person ' num2str(personer(i))]},'FontSize',10);
148     xlabel('Tid[s]')
149     ylabel('Amplitude')
150     set(gcf, 'Position', [2575, 400, 600, 600]);  % Set size
151 end
152 if savefigures  % Lagrer figuren
153     saveFigures('MiksedeSignaler',eksperiment)
154 end
155 
156 
```

```
% Separert lydopptak
figure(3),clf
n = size(U,1);
for i = 1:n%n:-1:1
    subplot(n,1,i)
    plot(linspace(0,length(U(i,:))/Fs,length(U(i,:))),U(i,:))
    if i==1
        title(['Separerte lydopptak'],'FontSize',16,'FontWeight','bold')
    end
    legend({['u_' num2str(i) '(n), opptak ' num2str(opptak(i))...
        ', person ' num2str(personer(i))]},'FontSize',10);
    xlabel('Tid[s]')
    ylabel('Amplitude')
    set(gcf, 'Position', [2575, 400, 600, 600]);  % Set size
end
if savefigures  % Lagrer figuren
    saveFigures('SeparerteSignaler',eksperiment)
end



% Brukes til   lagre plot med fornuftige navn
function saveFigures(name_,eksperiment)
n = 1;
[¬,img_dir] = dir_master();

formatData1 = 'epsc';
formatData2 = 'png';
if strcmp(eksperiment,'a')
    n = 1;
elseif strcmp(eksperiment,'b')
    n = 2;
end

name = strcat(name_,'_',num2str(n));
savedir = strcat(img_dir,'BSS\',name);


saveas(gcf,savedir,formatData1) % epsc
saveas(gcf,savedir,formatData2) % png
```

```
197 disp([name_ ' er lagret'])
198 end
```

8.2.9 Eksperiment 9

```
% Eksperiment 9: Template Matching

% Master new
% Master main

%% Clear, initialize
clear,clc;
[data_dir,img_dir] = dir_master();
warning off
addpath('Funksjoner/')

%% Load data


%  Definerer bndpassfilter
filtA = designfilt('bandpassiir', 'StopbandFrequency1', 1000,...
    'PassbandFrequency1', 2000, 'PassbandFrequency2', 15000,...
    'StopbandFrequency2', 16000, 'StopbandAttenuation1', 40,...
    'PassbandRipple', 1, 'StopbandAttenuation2', 40, 'SampleRate',...
    48000, 'DesignMethod', 'ellip');


saveData = 0;

folders = {'Opptak05','Opptak06','Laerdal'};

for i = 1:length(folders)
    if strcmp(folders{i},'Laerdal')
        recordNums{i} = 1:14;
        personNums{i} = 1:8;
    elseif strcmp(folders{i},'Opptak05')
        recordNums{i} = 1:8;
        personNums{i} = 1:5;
    elseif strcmp(folders{i},'Opptak06')
        recordNums{i} = 1:6;
        personNums{i} = 1:5;
```

```
37      end
38  end
39
40  % Fjern % for   kun laste og vise resultatert
41  for i = 1:length(folders)
42      recordNums{i} = [];
43      personNums{i} = [];
44  end
45
46
47  Names = {'clickAnn','cutpos','result_template','trueCanals',...
48      'detectedClick','maxValue'};
49
50
51
52  for f = 1:length(folders)
53      mappe = folders{f};
54      files = dir(strcat(data_dir,mappe,'\'));
55      [fileNames,R,P,N] = findFiles(files);
56
57      [clickAnn,cutpos,result_template,annoterteKanaler,...
58          Det,maksVerdi] = loadData(mappe,Names);
59
60
61      for r = recordNums{f}
62          for p = personNums{f}
63
64              %% Last inn lydopptak
65              filnavn = fileNames{r,p}
66              [x,Fs] = audioread(strcat(data_dir,mappe,'\',filnavn));
67              delayFirst = Fs;
68
69
70              x = x(cutpos(r,p)+delayFirst:end,:);% Synkroniseringstidspunkt
71
72              % Resampler dersom ndvendig
73              if ¬(Fs == 48000)
74                  x = resample(x,48000,Fs);
75                  Fs = 48000;
76              end
```

```
77              % (1) Initialisering
78              [¬,detektertKanal,kuttid,toppunktMin,Det_amp,maksVerdi_,mal]...
79                  = initialisering(x,filtA,filnavn,mappe,annoterteKanaler,1);
80
81              % Fjern % for   bruke annoterte kanaler istendfor detekterte
82              %             bestChannel = trueCanals(r,p);
83
84
85              % Lydopptaket frem til kuttid brukes i initialisering, resten
86              % av lydopptaket brukes videre.
87              x = x(kuttid:end,detektertKanal)';
88
89
90
91
92
93              %% Definerer parametre
94
95              dobbelKlikkDistanse = 0.4*Fs; % Avstand mellom to kompresjoner
96              secondsUsed = 3;% Forrige antall sekunder for   finne maksVerdi
97              sl = 18000;         % Signal lengde p  vindu
98              delay = 6000;       % Lag
99              overlap = 12000;    % Overlap p  vinduet
100             terskelFaktor = 0.6; % Forhold mellom maksVerdi og terskelVerdi
101             sampler = 100;      % Minste avstand mellom toppunkt i envelope
102             % sampler = 6000;  % Used to find envelope
103
104
105
106             x_bp = [];  % B ndpassfiltrert  signal
107
108             absoluttMaks = maksVerdi_;  % Initiell maksverdi
109             maksVerdi{r,p} = maksVerdi_;    % Maksverdi
110             terskelVerdi = maksVerdi_*0.7;  % Terskelverdi
111             x_env = [];     % Envelope
112             pks = [];       % Toppunkt over terskelverdi
113             locs = [];      % med tilh rende lokasjon
114             i = 1;          % Indeks
115             x_full = x;     % x_full er hele lydopptaket
116             x = [];         % Sletter x
```

```
117
118             while 1
119                 % Et vindu vil skyves over lydopptaket og det vil bare v re
120                 % tilgjengelig et vindu p  18000 sampler som forskyves 6000
121                 % sampler om gangen. Dermed blir det overlapp p  12000
122                 % sampler
123
124                 % Stopper enden av lydopptaket  n s
125                 if i+delay+sl≥length(x_full)
126                     break;
127                 end
128
129
130                 % Initiell
131                 if i==1
132                     % x(n) lydsignal
133                     x(1:2*delay+overlap) = x_full(1:2*delay+overlap);
134                     % x_bp(n)  B ndpassfiltrert
135                     x_bp(1:2*delay+overlap) = filter(filtA,x);
136
137                     x_env(1:delay+overlap) = zeros(1,delay+overlap);
138                     [x_env_,¬] = envelope(x_bp(1:2*delay+overlap)...
139                         ,sampler,'peak');
140                     x_env(1:delay+overlap) = x_env_(1:delay+overlap);
141                 end
142                 i = length(x);
143
144                 % Forskyver vinduet 6000 sampler
145                 x(i+1:i+sl-overlap) = x_full(i+1:i+sl-overlap);
146                 i = length(x)-delay;    % Indeks
147
148                 %% (2) Filtrer
149                 % y_filtrert og y ligger p  tiden (real time):
150                 x_bp(i+delay-sl+1:i+delay) = (filter(filtA,...
151                     x(i+delay-sl+1:i+delay)));
152
153
154                 %% (3) Envelope
155                 % alt det andre ligger 1 delay bak.
156                 [x_env_,¬] = envelope(x_bp(i-delay-sl+1:i+delay),...
```

```
157                     sampler,'peak');
158
159                 x_env(i-sl+1:i) = x_env_(delay+1:delay+sl);
160
161
162
163                 %% (4) Konvolver
164                 x_conv = conv(x_env(i-sl+1:i),mal,'same');    % Convolution
165
166
167                 %% (5) Oppdater terskelVerdi
168
169                 % Dersom det er toppunkt over terskelverd...
170                 if max(x_conv) ≥ terskelVerdi(end)
171
172                     % Toppunkt siste 3 sekund
173                     toppunkt = sort(pks(find(locs>i-secondsUsed*Fs)),...
174                         'descend');
175
176                     if length(pks)<2 % Initell
177                         maksVerdi{r,p}(1:i) = maksVerdi_;
178                     elseif length(toppunkt)>1
179                         maksVerdi{r,p}(i-sl+1:i) = max(mean(toppunkt(...
180                             1:min(length(toppunkt),...
181                             round(secondsUsed*100/60)))),absoluttMaks*0.4);
182
183                     elseif length(toppunkt)≤1
184                         maksVerdi{r,p}(i-sl+1:i) = max(maksVerdi{r,p}...
185                             (end)-0.1*absoluttMaks,absoluttMaks*0.4);
186                     end
187
188                     terskelVerdi(end+1:i) = maksVerdi{r,p}(end)*...
189                         terskelFaktor;
190
191                     %% (6) Detekter toppunkt
192                     [M,I] = max(x_conv);
193                     pks(end+1) = M;      % Amplitude til tidligere toppunkt
194                     locs(end+1) = i-sl+I;    % og lokasjon
195
196                     % Oppdaterer absolutMaks
```

```
197                     absoluttMaks = max(pks);
198
199                     click_posisjon = locs(end);
200
201                     %% (7) Detekter godkjente kompresjoner
202                     if click_posisjon-Det{r,p}(end)>dobbelKlikkDistanse
203                         Det{r,p}(end+1) = click_posisjon;
204                     elseif Det_amp(end)<pks(end)
205                         Det{r,p}(end) = click_posisjon;
206                     end
207
208                 else    % Bruker samme maksVerdi og terskelVerdi videre
209                     maksVerdi{r,p}(i-sl+1:i) = maksVerdi{r,p}(end);
210                     terskelVerdi(end+1:i) = terskelVerdi(end);
211                 end
212             end
213             Det{r,p}(1) = [];
214
215
216             %% Lagrer resultater i result_template fil
217             [result_template] = compareClick(clickAnn,Det{r,p},r,p,...
218                 x_env,result_template);
219
220
221             %% Save data
222             if saveData
223                 save(strcat(data_dir,mappe,'\','result_template.mat'),...
224                     'result_template')
225                 disp('result_template er lagret')
226             end
227
228         end
229
230     end
231
232
233     %% Resultater
234     visResultater = 0; % Vis tAcc, PPV, NPV osv i command window
235     [tAcc,PPV,NPV,Sen,Spe,tAccMat,PPVMat,NPVMat,SenMat,SpeMat,results] =...
236         makeConfusion(result_template,visResultater,mappe);
```

```
237
238     % Forvirringsmatrise (confusion matrix)
239     conf = {results(1,1),results(1,2);
240         results(2,1),results(2,2)};
241
242     % Cellestruktur med resultater til bruk i tabell
243     mainMat{1,f} = tAcc*100;
244     mainMat{2,f} = PPV*100;
245     mainMat{3,f} = NPV*100;
246     mainMat{4,f} = Sen*100;
247     mainMat{5,f} = Spe*100;
248     mainMat{6,f} =  conf;
249
250
251 end
252
253 %% Genererer tabell som skrives ut i command window
254 leftTitle = {'tAcc ','PPV','NPV','Sen','Spe',...
255     '\thead{Forvirrings- \\ matrise}'};
256
257 topTitle = {'','Datasett 5','Datasett 6','Datasett Laerdal'};
258 caption = 'Resultater fra \textit{template matching}';
259 label = 'tab:resultatTemplateMatching';
260 tabGen(mainMat,topTitle,leftTitle,4,caption,label,6);
```

8.2.10 Eksperiment 10

```
% Eksperiment 10: Adaptiv amplitude terskling


%% Clear, initialize
clear; clc; clear sound;
addpath('Funksjoner/')
[data_dir] = dir_master();
warning off



%  Definerer b ndpassfilter
filtA = designfilt('bandpassiir', 'StopbandFrequency1', 1000,...
    'PassbandFrequency1', 2000, 'PassbandFrequency2', 15000,...
    'StopbandFrequency2', 16000, 'StopbandAttenuation1', 40,...
    'PassbandRipple', 1, 'StopbandAttenuation2', 40, 'SampleRate',...
    48000, 'DesignMethod', 'ellip');



folders = {'Opptak05','Opptak06','Laerdal'};

for i = 1:length(folders)
    if strcmp(folders{i},'Laerdal')
        recordNums{i} = 1:14;
        personNums{i} = 1:8;
    elseif strcmp(folders{i},'Opptak05')
        recordNums{i} = 1:8;
        personNums{i} = 1:5;
    elseif strcmp(folders{i},'Opptak06')
        recordNums{i} = 1:6;
        personNums{i} = 1:5;
    end
end

% Fjern % for   kun laste og vise resultatert
```

```
% for i = 1:length(folders)
%     recordNums{i} = [];
%     personNums{i} = [];
% end



saveData = 1;

% [1] ved bruk av annoterte kanaler, [0] ved bruk av detekterte kanaler
brukAnnoterteKanaler = 1;


if brukAnnoterteKanaler
    Names = {'clickAnn','cutpos','result_trueChannels','trueCanals',...
        'detectedClick','maxValue'};
else
    Names = {'clickAnn','cutpos','result','trueCanals',...
        'detectedClick','maxValue'};
end




for f = 1:length(folders)
    mappe = folders{f};
    [clickAnn,cutpos,result,annoterteKanaler,Det,maksVerdi] =...
        loadData(mappe,Names);

    files = dir(strcat(data_dir,mappe,'\'));
    [fileNames,R,P,N] = findFiles(files);




    for r = recordNums{f}
        for p = personNums{f}

            filename = fileNames{r,p}
            [x,Fs] = audioread(strcat(data_dir,mappe,'\',filename));
```

```
77              delayFirst = Fs;
78
79              x = x(cutpos(r,p)+delayFirst:end,:);% Synkroniseringstidspunkt
80
81              % Resampler dersom ndvendig
82              if ¬(Fs == 48000)
83                  x = resample(x,48000,Fs);
84                  Fs = 48000;
85              end
86              %% (1) Initialisering
87              [¬,bestChannel,cutTime,peakMin_,Det_amp_,maksVerdi_]...
88                  =initialisering(x,filtA,filename,mappe,annoterteKanaler,0);
89
90              if brukAnnoterteKanaler
91                  bestChannel = annoterteKanaler(r,p);
92              end
93
94              % Lydopptaket frem til kuttid brukes i initialisering, resten
95              % av lydopptaket brukes videre.
96              x = x(cutTime:end,bestChannel)';
97
98
99
100             %% Definerer parametre
101
102             dobbelKlikkDistanse = 0.4*Fs; % Avstand mellom to kompresjoner
103
104             enkelKlikkDistanseMaks = 0.25*Fs;% Avstand mellom to klikk maks
105             enkelKlikkDistanseMin = 0.05*Fs; % Avstand mellom to klikk min
106
107             Det{r,p} = 0;       % Posisjon til detektert kompresjon og
108             % og tilhrende amplitude, gjennomsnitt av amplituden
109             % til klikk ned og klikk opp
110             Det_amp = Det_amp_;
111
112             toppunktMin = peakMin_; % Minste amplitude, klikk ned eller opp
113
114             % Minste avstand mellom to toppunkt, brukes i findpeaks
115             toppunktDistanse = 500;
116             sampler = 100;    % Avstand mellom toppunkt, brukes i envelope
```

```
117
118             % Forrige antall sekunder som brukes til   finne maksVerdi
119             secondsUsed = 3;
120
121             sl = 6000;          % Vindusstrrelse, signallengde
122             delay = sl;         % Lag
123             terskelFaktor = 0.3; % Forhold mellom maksVerdi og terskelVerdi
124             absoluttMax = maksVerdi_;
125
126
127
128             x_bp = [];  % B ndpassfiltrert  signal
129             maksVerdi{r,p} = maksVerdi_;     % Maksverdi
130             terskelVerdi = 0.3*maksVerdi_;   % Terskelverdi
131             x_env = [];     % Envelope
132             pks = [];       % Toppunkt over terskelverdi
133             locs = [];      % med tilhrende lokasjon
134             i = 1;          % Indeks
135
136             x_full = x;     % x_full er hele lydopptaket
137             x = [];         % Sletter y
138
139             while 1
140                 % Et vindu vil skyves over lydopptaket og det vil bare v re
141                 % tilgjengelig et vindu p  6000 sampler som forskyves 6000
142                 % sampler om gangen.
143
144                 % Stopper enden av lydopptaket  n s
145                 if i+delay+sl≥length(x_full)
146                     break;
147                 end
148
149                 % Initiell
150                 if i==1
151                     % x(n) lydsignal
152                     x(1:2*delay) = x_full(1:2*delay);
153
154                     % x_bp(n)  B ndpassfiltrert
155                     x_bp(1:2*delay) = filter(filtA,x);
156
```

```
157                     % Envelope
158                     [x_env_,¬] = envelope(x_bp(1:2*delay),sampler,'peak');
159                     x_env(1:delay) = x_env_(1:delay);
160
161                 end
162
163                 i = length(x);
164
165                 % Forskyver vinduet 6000 sampler
166                 x(i+1:i+sl) = x_full(i+1:i+sl);
167
168                 i = length(x)-delay;    % Indeks
169
170
171                 %% (2) Filtrer
172                 % y_filtrert og y ligger i sanntid (real time):
173                 x_bp(i+delay-sl+1:i+delay) = (filter(filtA,...
174                     x(i+delay-sl+1:i+delay)))/normFactor;
175
176
177                 %% (3) Envelope
178                 % alt det andre ligger 1 delay bak.
179                 [x_env_,¬] = envelope(x_bp(i-delay-sl+1:i+delay),...
180                     sampler,'peak');
181
182                 x_env(i-sl+1:i) = x_env_(delay+1:delay+sl);
183
184
185
186
187                 %% (4) Oppdater terskelVerdi
188
189                 % Toppunkt siste 3 sekund
190                 toppunkt =sort(pks(find(locs>i-secondsUsed*Fs)),'descend');
191
192                 if length(pks)<2 % Initell
193                     maksVerdi{r,p}(1:i) = maksVerdi_;
194
195                 elseif length(toppunkt)>1
196                     maksVerdi{r,p}(i-sl+1:i) = max(mean(toppunkt(...
```

```
197                         1:min(length(onlyPeaks),...
198                         round(secondsUsed*100/60*2)))),absoluttMax*0.4);
199 
200                 elseif length(toppunkt)≤1
201                     maksVerdi{r,p}(i-sl+1:i) = max(maksVerdi{r,p}(end)...
202                         -0.1,absoluttMax*0.5);
203                 else
204                     disp('Test')
205                 end
206 
207 
208                 terskelVerdi(end+1:i) = maksVerdi{r,p}(end)*terskelFaktor;
209 
210 
211 
212                 %% (5) Detekter toppunkt
213 
214                 % Finner toppunkt innenfor vinduet p  6000 sampler
215                 [pks_,locs_]= findpeaks(x_env(i-sl+1:i),'MinPeakHeight',...
216                     terskelVerdi(end),'MinPeakDistance',toppunktDistanse);
217 
218                 pks(end+1:end+length(pks_)) = pks_;          % Amplitude og
219                 locs(end+1:end+length(locs_)) = locs_+i-sl; % posisjon
220 
221                 % Oppdaterer absolutMaks
222                 absoluttMax = max(pks);
223 
224 
225 
226 
227                 %% (6) Detekter godkjente kompresjoner
228                 % Finner toppunkt over terskelverdi i sl (6000sampler) +
229                 % enkelKlikkDistanseMaks (19200sampler) tidligere sampler:
230                 index = find(locs>i-enkelKlikkDistanseMaks-sl);
231                 sl_Peaks = pks(index);
232                 sl_Locs = locs(index);
233 
234                 % Sorterer funnet toppunkt, Tar utgangspunkt i B(1) og I(1)
235                 % som er toppunkt med hyest amplitude innenfor vinduet.
236                 % Antar dett tilhrer et klikk og ser om man finner
```

```
237                 % tilhrende klikk.
238                 [B,I] = sort(sl_Peaks,'descend');
239
240                 % Finner toppunkt med hyeste amplitude, mer enn
241                 % toppunktMin - 0.1*absoluttMaks
242                 if length(B)>1 && sl_Peaks(I(1))>mean(toppunktMin)...
243                         -0.1*absoluttMaks
244
245                     % posLocs1 = mulig tilhrende klikk til venstre, Finner
246                     % toppunkt innenfor gult omrde til venstre i figur 4.8
247                     posLocs1 = find((locs>sl_Locs(I(1))-...
248                         enkelKlikkDistanseMaks) .* (locs<sl_Locs(I(1))...
249                         -enkelKlikkDistanseMin));
250
251                     % posLocs2 = mulig tilhrende klikk til hyre, Finner
252                     % toppunkt innenfor gult omrde til hyre i figur 4.8
253                     posLocs2 = [];
254                     if i>sl_Locs(I(1))+enkelKlikkDistanseMaks
255                         posLocs2 = find((locs<sl_Locs(I(1))...
256                             +enkelKlikkDistanseMaks) .* ...
257                             (locs>sl_Locs(I(1))+enkelKlikkDistanseMin));
258                     end
259
260                     % Lager vektor med mulige klikk ned og mulige klikk opp
261                     posLocs = [posLocs1 posLocs2];
262
263
264                     % Finner lokasjon til toppunkt med hyest amplitude
265                     [¬,posLoc_] = max(pks(posLocs));
266                     posLoc = posLocs(posLoc_);
267
268
269
270                     if length(posLoc)≥1 % Funnet tilhrende toppunkt?
271
272                         % Finner posisjon og amplitude til tilhrende klikk
273                         scndClickLocs = locs(posLoc);
274                         scndClickPks = pks(posLoc);
275
276                         % Finner posisjon til kompresjon
```

```
277                         kompPosisjon = mean([sl_Locs(I(1)) scndClickLocs]);
278
279                         % Gjennomsnittlig amplitude til mulig kompresjon:
280                         peakMean_ = mean([sl_Peaks(I(1)) scndClickPks]);
281
282                         % Avstand mellom Det(i) og Det(i-1) mer enn
283                         % dobbelKlikkDistanse:
284                         if kompPosisjon-Det{r,p}(end)>dobbelKlikkDistanse
285                             % Ja: lagre som godkjent kompresjon
286                             Det{r,p}(end+1) = kompPosisjon;
287                             toppunktMin(end+1) = min(sl_Peaks(I(1)),...
288                                 scndClickPks);
289
290                             Det_amp(end+1) = peakMean_;
291
292                         elseif Det_amp(end)<peakMean_
293                             % St rre gjennomsnittlig amplitude p  klikk ned
294                             % og klikk opp p  denne "mulige" kompresjonen
295                             % enn forrige godkjente kompresjon (Det(i)).
296                             % Erstatter da forrige kompresjon med denne.
297                             Det{r,p}(end) = kompPosisjon;
298                             toppunktMin(end) = min(sl_Peaks(I(1)),...
299                                 scndClickPks);
300
301                             Det_amp(end) = peakMean_;
302
303                         end
304
305
306                     end
307                 end
308
309             end
310
311             % Sletter f rste kompresjon fra initialiseringen
312             Det{r,p}(1) = [];
313
314
315             % Sammenligner detekterte kompresjoner med annoterte
316             % kompresjoner og lager en cellestrukter med
```

```
317             % forvirringsmatriser (Confusion Matrix)
318             [result] = compareClick(clickAnn,Det{r,p},r,p,...
319                 x_env,result);
320
321
322             %% Save data
323             if saveData
324                 maxValue = maksVerdi;
325                 save(strcat(data_dir,folder,'\','maxValue.mat'),'maxValue')
326                 disp('maxValue er lagret')
327                 if brukAnnoterteKanaler
328                     result_trueChannels = result;
329                     save(strcat(data_dir,mappe,'\',...
330                         'result_trueChannels.mat'),'result_trueChannels')
331                     disp('result_trueChannels er lagret')
332                 else
333                     save(strcat(data_dir,mappe,'\','result.mat'),'result')
334                     disp('result er lagret')
335                 end
336                 detectedClick = Det;
337                 save(strcat(data_dir,folder,'\','detectedClick.mat'),...
338                     'detectedClick')
339                 disp('detectedClick er lagret')
340
341             end
342
343         end
344
345     end
346
347     %% Resultater
348     visResultater = 0; % Vis tAcc, PPV, NPV osv i command window
349
350     [tAcc,PPV,NPV,Sen,Spe,¬,¬,¬,¬,¬,results] = ...
351         makeConfusion(result,visResultater,mappe);
352
353     conf = {results(1,1),results(1,2);
354         results(2,1),results(2,2)};
355
356     mainMat{1,f} = tAcc*100;
```

```
357     mainMat{2,f} = PPV*100;
358     mainMat{3,f} = NPV*100;
359     mainMat{4,f} = Sen*100;
360     mainMat{5,f} = Spe*100;
361     mainMat{6,f} =  conf;
362
363
364 end
365
366 leftTitle = {'tAcc ','PPV','NPV','Sen','Spe',...
367     '\thead{Forvirrings- \\ matrise}'};
368
369 topTitle = {'','Datasett 5','Datasett 6','Datasett Laerdal'};
370 caption = 'Resultater fra Adaptiv Amplitude Terskling';
371 label = 'tab:resultatAdaptivAmplitude';
372 tabGen(mainMat,topTitle,leftTitle,4,caption,label,6);
```

8.2.11 Eksperiment 11

```
% Eksperiment 11: Paavirkning fra nabodukker

% findClosePeaks
% Script for aa finne ut hvor hoye n rliggende peaks er.


%% Clear, initialize
clear; clc; clear sound;
addpath('Functions/')
[data_dir,img_dir] = dir_master();
warning off

%% Laster data

% Genererer filter
filtA = designfilt('bandpassiir', 'StopbandFrequency1', 1000,...
    'PassbandFrequency1', 2000, 'PassbandFrequency2', 15000,...
    'StopbandFrequency2', 16000, 'StopbandAttenuation1', 40,...
    'PassbandRipple', 1, 'StopbandAttenuation2', 40, 'SampleRate',...
    48000, 'DesignMethod', 'ellip');

Fs = 48000; % Starten av lydopptaktet er 1 sekund etter klapping

% Velg enten datasett 5 eller 6
folder = 'Opptak05';% Datasett
records = 8;
persons = 5;

% folder = 'Opptak06';
% records = 6;
% persons = 5;


personNum = 3;
recordNum = 1:records;

```

```
37
38 files = dir(strcat(data_dir,folder,'\'));
39 [fileNames,¬,¬,¬] = findFiles(files);
40 col = {'r','g','b','c','m'};
41
42
43 for r = recordNum
44     for p = personNum
45
46         figs2keep = [1, 2, 3, 4];
47         all_figs = findobj(0, 'type', 'figure');
48         if length(all_figs) > 4
49             try
50                 delete(setdiff(all_figs, figs2keep));
51             catch
52                 disp('Ikke alle figurene er oppe')
53             end
54         end
55
56         % y_upper = x_env(n), envelopen til x(n)
57         % clickAnn = annoterte kompresjoner
58         % trueRightClick = annotert klikk opp
59         % trueLeftClick = annotert klikk ned
60         Names = {'clickAnn','y_upper','trueRightClick','trueLeftClick'};
61         if ¬(exist('clickAnn'))  % Laster data dersom det trengs
62             [clickAnn,x_env,annotertKlikkOpp,annotertKlikkNed]...
63                 = loadData(folder,Names);
64         end
65
66         [pks,locs] = findpeaks(x_env{r,p}); % Finner toppunkt
67
68         figure(1),clf
69         plot(linspace(0,length(x_env{r,p})/Fs,length(x_env{r,p})),...
70             x_env{r,p},'k') % Plotter envelopen
71         hold on
72
73         % Alle annoterte klikk ned og klikk opp plottes paa hoyeste toppunkt
74         % paa envelopen til person 3 forskjovet med maks 300 sampler. Dette
75         % paa grunn av at det er noen sampler forskyvelse i tid mellom
76         % opptakene. Envelopen kan ogsaa gi toppunkt som er forskjovet noen
```

```
77          % sampler.
78          distMove = 300;
79          otherPersons = 1:persons;
80          otherPersons(find(otherPersons==p)) = [];
81          for i = 1:persons
82              if i == p
83                  sumOfAll(r,i) = 0;
84                  scatter([annotertKlikkOpp{r,p},annotertKlikkNed{r,p}]/Fs,...
85                      [x_env{r,p}(annotertKlikkOpp{r,p}),...
86                      x_env{r,p}(annotertKlikkNed{r,p})],'y')
87              else
88
89                  leftPeak = [];
90                  rightPeak = [];
91                  lValue = [];
92                  rValue = [];
93                  for k = 1:length(annotertKlikkNed{r,i})
94                      leftPeak(end+1) = finnPeaks(pks,locs,...
95                          annotertKlikkNed{r,i}(k),distMove);
96
97                      checkDist = min(abs(annotertKlikkNed{r,p}-...
98                          leftPeak(end)));
99
100                     if checkDist<20
101                         if length(lValue)>0
102                             lValue(end+1) = round(mean(lValue));
103                         else
104                             lValue(end+1) = 0;
105                         end
106
107                     else
108                         lValue(end+1) = x_env{r,p}(leftPeak(end));
109                     end
110
111                     rightPeak(end+1) = finnPeaks(pks,locs,...
112                         annotertKlikkOpp{r,i}(k),distMove);
113
114                     checkDist = min(abs(annotertKlikkOpp{r,p}-...
115                         rightPeak(end)));
116
```

```
117                     if checkDist<20
118                         if length(rValue)>0
119                             rValue(end+1) = round(mean(rValue));
120                         else
121                             rValue(end+1) = 0;
122                         end
123                     else
124                         rValue(end+1) = x_env{r,p}(rightPeak(end));
125                     end
126
127
128
129
130                 end
131                 % Sum av amplituden
132                 sumOfAll(r,i)=sum([rValue,lValue]/length([rValue,lValue]));
133
134                 % Plotter toppunkt til klikk fra andre dukker.
135                 scatter([leftPeak,rightPeak]/Fs,[x_env{r,p}(leftPeak),...
136                     x_env{r,p}(rightPeak)],col{i})
137             end
138         end
139
140         scatter([annotertKlikkOpp{r,p},annotertKlikkNed{r,p}]/Fs,...
141             [x_env{r,p}(annotertKlikkOpp{r,p}),...
142             x_env{r,p}(annotertKlikkNed{r,p})],'y')
143
144         if strcmp(folder,'Opptak05')
145             title(['Forurensning fra andre dukker, Datasett 5, opptak: '...
146                 num2str(r) ', person: ' num2str(p)],'FontSize',16,...
147                 'FontWeight','bold')
148             tittel = 'GjNaboAplitudeOpptak5';
149         elseif strcmp(folder,'Opptak06')
150             title(['Forurensning fra andre dukker, Datasett 6, opptak: '...
151                 num2str(r) ', person: ' num2str(p)],'FontSize',16,...
152                 'FontWeight','bold')
153             tittel = 'GjNaboAplitudeOpptak6';
154         end
155         legend({'x_{env}(n)','Klikk dukke 1','Klikk dukke 2',...
156             'Klikk dukke 3','Klikk dukke 4','Klikk dukke 5'},'FontSize',12)
```

```
157         xlabel('Tid[s]')
158         ylabel('Amplitude')
159         axis([0 23 -0.2 1.1])
160         set(gcf, 'Position', [2575, 700, 1000, 600]);  % Set size
161         % saveFigs(['Forurensning' num2str(r) '_' num2str(p)],gcf,2,1,...
162         % [2575, 700, 1000, 600])
163
164
165         drawnow
166         % saveFigs(['Forurensning' num2str(r) '_' num2str(p)],gcf,4)
167
168     end
169 end
170 save(strcat(data_dir,folder,'\','sumOfAll.mat'),'sumOfAll')
171 disp('sumOfAll er lagret')
172
173 if strcmp(folder,'Opptak05')
174     rekkefolge = [1 2 3 4 7 5 8 6];
175 elseif strcmp(folder,'Opptak06')
176     rekkefolge = [1 2 3 4 5 6];
177 end
178
179 sumOfAll = sumOfAll(rekkefolge,:); % Sortert paa lengde
180
181
182
183 % sumOfAll = sumOfAll([1 4 5 6 2 7 3 8],:);   % Sortert paa hoyde
184 figure(2),clf
185 M = size(sumOfAll,1);
186 N = size(sumOfAll,2);
187
188 plot(sumOfAll')
189 if strcmp(folder,'Opptak05')
190     title(['Paavirkning fra nabodukker: Datasett 5, person ' num2str(p)],...
191         'FontSize',16,'FontWeight','bold')
192     tittel = 'GjNaboAplitudePerson5';
193 elseif strcmp(folder,'Opptak06')
194     title(['Paavirkning fra nabodukker: Datasett 6, person ' num2str(p)],...
195         'FontSize',16,'FontWeight','bold')
196     tittel = 'GjNaboAplitudePerson6';
```

```
197 end
198 legend({'Opptak 1','Opptak 2','Opptak 3','Opptak 4','Opptak 5',...
199     'Opptak 6','Opptak 7','Opptak 8'},'FontSize',12)
200 xlabel('Person')
201 ylabel(['Gjennomsnitlig amplitude paa toppunkt tilhorende '...
202     'klikk fra nabodukker'])
203 set(gcf, 'Position', [2575, 300, 1000, 600]);  % Set size
204
205 % saveFigs(tittel,gcf,2,1,[2575, 300, 1000, 600])
206
207
208
209 %% Paavirkning fra nabodukker, pr opptak
210 figure(3),clf
211 sumfig3 = mean(sumOfAll(:,[1 2 4 5]),2);
212 % sumfig3 = sumOfAll(:,4);
213 % sumfig3 = [sumOfAll(:,[1 2 4 5]) mean(sumOfAll,2)];
214 plot(sumfig3)
215 hold on
216 scatter(1:length(sumfig3),sumfig3,'k','LineWidth',2)
217 if strcmp(folder,'Opptak05')
218     title(['Paavirkning fra nabodukker, pr opptak: Datasett 5, person '...
219         num2str(p)],'FontSize',16,'FontWeight','bold')
220     tittel = 'GjNaboAplitudeOpptak5';
221     lengde = [0.8 0.8 0.8 1.0 1.2 1.4 1.0 1.2];
222     hoyde = [1.2 1.4 1.6 1.2 1.2 1.2 1.4 1.6];
223 elseif strcmp(folder,'Opptak06')
224     title(['Paavirkning fra nabodukker: Datasett 6, person ' num2str(p)],...
225         'FontSize',16,'FontWeight','bold')
226     tittel = 'GjNaboAplitudeOpptak6';
227     lengde = [0.8:0.2:1.8];
228     hoyde = [ones(1,length(lengde))*1.4];
229 end
230
231 for i = recordNum
232     if i==recordNum(max(recordNum))
233         text(recordNum(i)-1.1,sumfig3(i)+0.005,['H='...
234             num2str(hoyde(rekkefolge(i))) 'm, L='...
235             num2str(lengde(rekkefolge(i))) 'm'],...
236             'FontSize',11,'FontWeight','bold')
```

```
237
238     else
239         text(recordNum(i)+0.1,sumfig3(i)-0.001,['H='...
240             num2str(hoyde(rekkefolge(i))) 'm, L='...
241             num2str(lengde(rekkefolge(i))) 'm'],...
242             'FontSize',11,'FontWeight','bold')
243     end
244 end
245
246
247 legend({'Paavirkning pr opptak'},'FontSize',12)
248 xlabel('Opptak')
249 ylabel(['Gjennomsnitlig amplitude paa toppunkt tilhorende '...
250     'klikk fra nabodukker'])
251 set(gcf, 'Position', [2575, 700, 1000, 600]);  % Set size
252 % tittel = [tittel 'Sortert'];
253 % saveFigs(tittel,gcf,2,1,[2575, 700, 1000, 600])
254
255
256 %%
257 function [trueLoc] = finnPeaks(pks,locs,pos,distMove)
258
259 [M,I] = min(abs(locs-pos));
260 if M<distMove
261     trueLoc = locs(I);
262 else
263     trueLoc = pos;
264 end
265
266
267 end
```

8.2.12 Funksjon: dir_master.m

```
function [data_dir,img_dir,img_rapport,img_gjort] = dir_master()
% Funksjon med definert sti til mappe med datamateriell
% Variablename:
% data_dir      = sti til lagrete data
% img_dir       = sti til lagring av figurer
% img_rapport   = sti til lagring av figurer i rapport
% img_gjort     = sti til lagring av figurer i annen mappe

startPath = 'C:\Users\oyvin\Dropbox\Masteroppgave\Matlab';
% startPath = cd;

data_dir = strcat(startPath,'\Data\');
img_dir = strcat(startPath,'\Figures\');
img_rapport = ...
    strcat('C:\Users\oyvin\Dropbox\Apps\ShareLaTeX\Masteroppgave\Bilder\');
img_gjort = 'C:\Users\oyvin\Dropbox\Masteroppgave\Matlab\hva_er_gjort\';
end
```

8.2.13 Funksjon: findBestChannel.m

```
function [besteKanal] = findBestChannel(x,plotdata,annotertKanal)
%Denne funksjonen finner beste kanal basert paa amplituden til fem
%kompresjoner
% Inngang:
% x             = lydsignal
% plotdata      = [1] lag figur, [2] ikke lag figur
% annotertKanal = annotert kanal 1 eller 2, for sammenligning
% Utgang:
% bestKanal     = detektert kanal 1 eller 2


[data_dir,¬] = dir_master();
if ¬nargin  % Brukt til testing
    load(strcat(data_dir,'FunksjonsLoad\','findBestChannelVar'))
end

if size(x,2) == 1   % Dersom kun en kanal er tilgjengelig
    besteKanal = 1;
    disp('iPhone: bare en kanal tilgjengelig.')
    return;
end


x1 = x(:,1); % Kanal 1
x2 = x(:,2); % Kanal 2

maks1 = max(x1)/4; % Terskel 1
maks2 = max(x2)/4; % Terskel 2

% Finner toppunkt over terskel paa begge kanaler
[pks1,locs1] = findpeaks(x1,'MinPeakHeight',maks1,'MinPeakDistance',500);
[pks2,locs2] = findpeaks(x2,'MinPeakHeight',maks2,'MinPeakDistance',500);

% Sorterer toppunktene i synkende rekkefolge
[B1,¬] = sort(pks1,'descend');
[B2,¬] = sort(pks2,'descend');
```

```
37
38 % Bruker opptil 5 toppunkt med hoyest amplitude fra kanal 1
39 if length(B1)≤5
40     antallKlikk1 = length(B1);
41 else
42     antallKlikk1 = 5;
43 end
44
45 % Bruker opptil 5 toppunkt med hoyest amplitude fra kanal 2
46 if length(B2)≤5
47     antallKlikk2 = length(B2);
48 else
49     antallKlikk2 = 5;
50 end
51
52 % Finner gjennomsnittlig amplitude paa opptil fem hoyeste toppunkt
53 gjAmplitude1 = mean(B1(1:antallKlikk1));
54 gjAmplitude2 = mean(B2(1:antallKlikk2));
55
56 % Velger beste kanal som har hoyest gjennomsnittlig amplitud paa opptil fem
57 % hoyeste toppunkt
58 [¬,besteKanal] = max([gjAmplitude1 gjAmplitude2]);
59
60
61 % Sammenligner med annotert kanal
62 if nargin ==3
63     disp(['Beste mulige kanal valgt. Kanal: ' num2str(besteKanal)...
64         '. Annotert: ' num2str(annotertKanal)])
65 else
66     disp(['Beste mulige kanal valgt. Kanal: ' num2str(besteKanal)])
67 end
68
69
70
71 %% Lag figur med begge kanaler
72 if plotdata
73     figure
74     subplot(2,1,1)
75     plot(x1)
76     hold on
```

```
77      scatter(locs1,pks1)
78      title('y1')
79
80
81      subplot(2,1,2)
82      plot(x2)
83      hold on
84      scatter(locs2,pks2)
85      title('y2')
86      drawnow
87  end
88  end
```

8.2.14 Funksjon: findFiles.m

```
function [fileNames,R,P,N] = findFiles(files)
%Denne funksjonen finner videofiler i en mappe og lager en cellestruktur
%med opptak i rader og personer i kolonne.
% fileNames =   Cellstructur av filnavn til videoer
% R         =   Antall opptak (records)
% P         =   Antall personer
% N         =   Antall filer
% files     =   Struct av filer i en mappe ved bruk av dir()funksjon


if ¬nargin % Brukt til testing
    clear,clc
    close all
    [data_dir,¬] = dir_master();
    folder = 'FebKjeller';
    files = dir(strcat(data_dir,folder,'\'));
end

N = numel(files);   % Antall filer

I = [];
n = 1;
for i = 1:N
    if files(i).bytes>10000 % Filen maa vre en viss storrelse og
        if strcmp(files(i).name(end-2:end),'mp4')||...  % et visst format
                strcmp(files(i).name(end-2:end),'MOV')||...
                strcmp(files(i).name(end-2:end),'m4v')||...
                strcmp(files(i).name(end-2:end),'mov')

            recordNum(n,1:2) = (files(i).name(1:2));
            person(n,1:2) = (files(i).name(4:5));
            r = str2num(recordNum(n,1:2));
            p = str2num(person(n,1:2));

            fileNames{r,p} = files(i).name;
            n = n+1;
```

```
37          end
38      end
39  end
40
41  R = str2num(unique(recordNum,'rows')); % Opptak
42  R(find(R==0),:) = []; % Fjerner 0
43
44  P = str2num(unique(person,'rows')); % Personer
45  P(find(P==0),:) = []; % Fjerner 0
46
47
48
49  end
```

8.2.15 Funksjon: generateTemplate.m

```
1  function [ template ] = generateTemplate(avstand,hoydeToppunkt,bredde,plotFigur)
2  %Denne funksjonen genererer en mal (template) av to gaussfunksjoner basert
3  %paa input om bredde paa funksjonene, hoyde og avstand.
4  % Inn:
5  % avstand       = avstand mellom gaussfunksjoner
6  % bredde        = bredde paa gaussfunksjon
7  % hoydeToppunkt = hoyde paa gaussfunksjoner, [1,1] blir brukt
8  % plotFigur     = [1] plott figur, [0] ikke plot
9  % UT:
10 % template      = mal
11
12 if ¬nargin
13     hoydeToppunkt = [1 1];
14     avstand = 6000;
15     bredde = 2000;
16 elseif nargin == 1
17     hoydeToppunkt = [1 1];
18     bredde = 2000;
19 elseif nargin == 2
20     bredde = 2000;
21 elseif nargin == 3
22     plotFigur = 0;
23 end
24
25
26
27 w = gausswin(bredde)';  % Lager gauss vindu.
28 w = w-min(w);
29 w = w/max(w); %Normaliserer
30
31 N = length(avstand);
32
33 Fs = 48000;
34 templateLength = 0.3*Fs; % Lengde paa malen
35
36
```

```
for i = 1:N

    template_ = [w*hoydeToppunkt(1),zeros(1,avstand(i)-length(w)),...
        w*hoydeToppunkt(2)];

    template_ = [zeros(1,round((templateLength-length(template_))/2)),...
        template_, zeros(1,round((templateLength-length(template_))/2))];

    if length(template_)==templateLength
        template(i,:) = template_;
    else
        disp('Length of template_ don`t match templateLength')
        reminder = templateLength-length(template_);
        if reminder > 0
            template = [template_ zeros(1,abs(reminder))];
            disp(['Added ' num2str(reminder) ' samples'])
        else
            template_ = template_(1:templateLength);
            template(i,:) = template_;
            disp(['Removed ' num2str(reminder) ' samples'])
        end
    end
end

if plotFigur
    figure(1),clf
    plot(template)
end

end
```

8.2.16 Funksjon: initialisering.m

```
function [normFactor,bestChannel,cutTime,toppunktMin,...
    Det_amp,maxValue,template] = ...
    initialisering(y,filtA,filename,folder,trueCanals,useTemplate)
%Initialisering
% Inputs:
% y             = lydopptak
% filtA         = filter
% filename      = filnavn
% folder        = mappe
% trueCanals    = annoterte kanaler paa lydopptakene
% useTemplate   = 1 for template matching, 0 ellers
%
% Outputs:
% normFaktor    = normaliseringsfaktor
% bestChannel   = beste kanal
% cutTime       = tidsrom som 5 initielle kompresjoner utfores
% peakMin       = Gjennomsnitt av alle forste klikk
% peakMean      = Gjennomsnitt av alle forste og andre klikk (tall)
% vanligvis vektor med gjennomsnitt av hvert forste og andre klikk.
% maxValue      = Gjennomsnitt av alle forste og andre klikk (tall)
% template      = Mal, brukes i template matching


if ¬nargin % Brukt for testing
    load(strcat(data_dir,'FunksjonsLoad\','preProssesVar'))
end

recordNum = str2num(filename(1:2)); % Opptak
personNum = str2num(filename(4:5)); % Person


% cutTimes er tid hvor grensen til initialisering gaar. Lydopptak for denne
% brukes her i initialisering og resten brukes til aa detektere
% kompresjoner. Datasett Laerdal har ikke fem kompresjoner i starten av
% opptaket og vil derfor maatte bruke hele lydopptaket til initialisering
if strcmp(folder,'Opptak05')
```

```
37      cutTimes = [467036,4.9275e+05,4.4858e+05,491306,5.4244e+05,...
38          4.9855e+05,509432,4.5176e+05];
39      cutTime = cutTimes(recordNum);
40      y = y(1:cutTime,:);  % Kutter ut omraadet som brukes til initialisering
41      % Finner beste kanal:
42      [bestChannel] = findBestChannel(y,0,trueCanals(recordNum,personNum));
43      y = y(:,bestChannel);
44      y_filtered = filter(filtA,y); % Filtrerer
45      normFactor = max(y_filtered); % Normaliseringsfaktor blir ikke brukt
46
47  elseif strcmp(folder,'Opptak04')
48      cutTimes = [3.5e5,1,1,1,1,1,1,1];
49      cutTime = cutTimes(recordNum);
50      y = y(1:cutTime,:);  % Kutter ut omraadet som brukes til initialisering
51      % Finner beste kanal:
52      [bestChannel] = findBestChannel(y,0,trueCanals(recordNum,personNum));
53      y = y(:,bestChannel);
54      y_filtered = filter(filtA,y); % Filtrerer
55      normFactor = max(y_filtered); % Normaliseringsfaktor blir ikke brukt
56
57  elseif strcmp(folder,'Opptak06')
58      cutTimes = [4.25e5 5.5e5 4.5e5 5.5e5 5.5e5 4.5e5 1];
59      cutTime = cutTimes(recordNum);
60
61      if ¬(recordNum == 7)
62          y = y(1:cutTime,:);
63      end
64      [bestChannel] = findBestChannel(y,0,trueCanals(recordNum,personNum));
65      y = y(:,bestChannel);
66
67      y_filtered = filter(filtA,y); % Filtrerer
68      normFactor = max(y_filtered); % Normaliseringsfaktor blir ikke brukt
69
70  else
71      cutTime = 1;
72      [bestChannel] = findBestChannel(y,0,trueCanals(recordNum,personNum));
73      y = y(:,bestChannel);
74      y_filtered = filter(filtA,y);
75      normFactor = max(y_filtered);
76  end
```

```
77
78 % Her gjores en kompresjonsdeteksjon tilsvarende Adaptiv amplitude
79 % terskling bare at denne gjores paa hele lydsignalet samtidig.
80
81 % Definerer parametre:
82 Fs = 48000;
83 % Minste avstand mellom to toppunkt, brukes i findpeaks
84 toppunktDistanse = 500;
85 sampler = 100;    % Avstand mellom toppunkt, brukes i envelope
86
87 dobbelKlikkDistanse = 0.4*Fs; % Avstand mellom to kompresjoner
88 enkelKlikkDistanseMaks = 0.25*Fs;% Avstand mellom to klikk maks
89 enkelKlikkDistanseMin = 0.05*Fs; % Avstand mellom to klikk min
90 Det = 0;
91
92 detektertVenstreKlikk = 0;  % Klikk ned
93 detektertHoyreKlikk = 0;    % Klikk opp
94 Det_amp = 0;
95 toppunktMin = 0; % Minste amplitude, klikk ned eller opp
96
97
98 [x_env,¬] = envelope(y_filtered,sampler,'peak'); % Finner envelopen
99 x_env = x_env/max(x_env);   % Normaliserer
100
101 % Finner toppunkt over terskelverdi, her 0.25
102 [pks,locs] = findpeaks(x_env,'MinPeakHeight',0.25,'MinPeakDistance',...
103     toppunktDistanse);
104 pks = pks';
105 locs = locs';
106
107 % Sorterer toppunkt
108 [pksSorted,I] = sort(pks,'descend');
109 locsSorted = locs(I);
110
111 % Gaar gjennom hvert toppunkt, antar det er ett klikk og finner tilhorende
112 % klikk.
113 for i = 1:length(pksSorted)-1
114     % posLocs1 = mulig tilhorende klikk til venstre, Finner
115     % toppunkt innenfor gult omraade til venstre i figur 4.8
116     posLocs1 = find((locs>locsSorted(1)-enkelKlikkDistanseMaks) .*...
```

```
117         (locs<locsSorted(1)-enkelKlikkDistanseMin));
118
119     % posLocs2 = mulig tilhorende klikk til hoyre, Finner
120     % toppunkt innenfor gult omraade til hoyre i figur 4.8
121     posLocs2 = find((locs<locsSorted(1)+enkelKlikkDistanseMaks) .*...
122         (locs>locsSorted(1)+enkelKlikkDistanseMin));
123
124     % Lager vektor med mulige klikk ned og mulige klikk opp
125     posLocs = [posLocs1 posLocs2];
126
127     % Finner lokasjon til mulig tilhorende toppunkt med hoyest amplitude
128     [¬,posLoc_] = max(pks(posLocs));
129     posLoc = posLocs(posLoc_);
130
131
132     if length(posLoc)≥1  % Funnet tilhorende toppunkt?
133         % Finner posisjon og amplitude til tilhorende klikk
134         scndClickLocs = locs(posLoc);
135         scndClickPks = pks(posLoc);
136
137         % Finner posisjon til kompresjon
138         kompPosisjon = round(mean([locsSorted(1) scndClickLocs]));
139
140         % Gjennomsnittlig amplitude til mulig kompresjon:
141         Det_amp_ = mean([pksSorted(1) scndClickPks]);
142
143         % Avstand mellom Det(i) og Det(i-1) mer enn dobbelKlikkDistanse:
144         if abs(kompPosisjon-Det(end))>dobbelKlikkDistanse
145             % Ja: lagre som godkjent kompresjon, lagre baade forst og andre
146             % klikk
147             Det(end+1) = kompPosisjon;
148             toppunktMin(end+1) = min(pksSorted(1), scndClickPks);
149             Det_amp(end+1) = Det_amp_;
150             detektertVenstreKlikk(end+1)=min([locsSorted(1) scndClickLocs]);
151             detektertHoyreKlikk(end+1) =max([locsSorted(1) scndClickLocs]);
152         elseif Det_amp(end)<Det_amp_
153             % Storre gjennomsnittlig amplitude paa klikk ned
154             % og klikk opp paa denne "mulige" kompresjonen
155             % enn forrige godkjente kompresjon (Det(i)).
156             % Erstatter da forrige kompresjon med denne.
```

```
157             Det(end) = kompPosisjon;
158             toppunktMin(end) = min(pksSorted(1), scndClickPks);
159             Det_amp(end) = Det_amp_;
160             detektertVenstreKlikk(end)= min([locsSorted(1) scndClickLocs]);
161             detektertHoyreKlikk(end) = max([locsSorted(1) scndClickLocs]);
162         end
163
164
165     end
166
167
168
169     pksSorted(1) = [];
170     locsSorted(1) = [];
171
172
173 end
174
175 Det(1) = [];
176 detektertVenstreKlikk(1) = [];
177 detektertHoyreKlikk(1) = [];
178 Det_amp(1) = [];
179 toppunktMin(1) = [];
180
181 if 0
182     figure(1),clf
183     plot(yupper)
184     hold on
185     scatter(Det,zeros(1,length(detectedClick)))
186     scatter(detektertVenstreKlikk,yupper(detectedLeftClick))
187     scatter(detektertHoyreKlikk,yupper(detectedRightClick))
188 end
189
190
191
192
193
194
195 Det_amp = mean(Det_amp);
196 toppunktMin = mean(toppunktMin);
```

```
197 maxValue=mean([x_env(detektertVenstreKlikk)',x_env(detektertHoyreKlikk)']);
198
199
200
201
202
203
204
205 % Ved template matching konvolveres x_env med en template og man finner
206 % n rmeste toppunkt til Det, detekterte kompresjoner
207 if useTemplate
208     distPeaks = round(median(detektertHoyreKlikk-detektertVenstreKlikk));
209     hoydepeak = [1,1];
210     widthPeak = 2000;
211     [template] = generateTemplate(distPeaks,hoydepeak,widthPeak,0);
212
213     r = conv(x_env,template,'same'); % Konvolusjon
214
215     for i = 1:length(Det)
216         [¬,I] = max(r(Det(i) - 200:Det(i) + 200));
217
218         detectedClick2(i) = Det(i) + I - 201;
219     end
220     Det_amp = mean([r(detectedClick2)',r(detectedClick2)']);
221     toppunktMin =  r(min(detectedClick2));
222     maxValue = mean([r(detectedClick2)',r(detectedClick2)']);
223
224 end
225
226
227 if ¬nargin % Plot figur, brukt under test
228     figure(3),clf
229     plot(r)
230     hold on
231     scatter(detectedClick2,r(detectedClick2))
232 end
233
234 end
```

8.2.17 Funksjon: loadData.m

```
function [O1,O2,O3,O4,O5,O6,O7,O8,O9,O10,O11,O12,O13,O14,O15] = loadData(mappe,navn)
%Funksjon for aa laste inn .mat filer. F
% Inn:
% mappe     = Mappe hvor filene ligger
% navn      = Navn paa alle filer i cellestruktur
% Ut:
% O1:O15    = Alle variabler som lastes inn, lik antall celler i navn

[data_dir,¬] = dir_master();
if ¬nargin
    load(strcat(data_dir,'FunksjonsLoad\','loadDataVar'))
end


N = length(navn);
for i = 1:N
    if exist(strcat(data_dir,mappe,'\',navn{i},'.mat'))
        disp(['Laster ' navn{i} '.mat'])
        data = load(strcat(data_dir,mappe,'\',navn{i},'.mat'));
        switch i
            case 1
                O1 = data.(navn{1});
            case 2
                O2 = data.(navn{2});
            case 3
                O3 = data.(navn{3});
            case 4
                O4 = data.(navn{4});
            case 5
                O5 = data.(navn{5});
            case 6
                O6 = data.(navn{6});
            case 7
                O7 = data.(navn{7});
            case 8
                O8 = data.(navn{8});
```

```
37              case 9
38                  O9 = data.(navn{9});
39              case 10
40                  O10 = data.(navn{10});
41              case 11
42                  O11 = data.(navn{11});
43              case 12
44                  O12 = data.(navn{12});
45              case 13
46                  O13 = data.(navn{13});
47              case 14
48                  O14 = data.(navn{14});
49              case 15
50                  O15 = data.(navn{15});
51              otherwise
52                  error('Too many files to load, please update loadData function')
53          end
54      else
55          disp(['Lager ny ' navn{i} '.mat'])
56
57          switch i
58              case 1
59                  O1 = chooseNew();
60              case 2
61                  O2 = chooseNew();
62              case 3
63                  O3 = chooseNew();
64              case 4
65                  O4 = chooseNew();
66              case 5
67                  O5 = chooseNew();
68              case 6
69                  O6 = chooseNew();
70              case 7
71                  O7 = chooseNew();
72              case 8
73                  O8 = chooseNew();
74              case 9
75                  O9 = chooseNew();
76              case 10
```

```
77                  O10 = chooseNew();
78              case 11
79                  O11 = chooseNew();
80              case 12
81                  O12 = chooseNew();
82              case 13
83                  O13 = chooseNew();
84              case 14
85                  O14 = chooseNew();
86              case 15
87                  O15 = chooseNew();
88              otherwise
89                  error('Too many files to load, please update loadData function')
90          end
91      end
92
93
94
95  end
96
97
98  end
99
100 % Lag ny fil
101 function O = chooseNew()
102
103 r = input('Tast inn antall opptak: ');
104 p = input('Tast inn antall personer: ');
105 while 1
106     c = input('[1] cellestruktur, [2] matrise struktur: ');
107
108     if c == 1
109         O{r,p} = [];
110         break;
111     elseif c == 2
112         O = zeros(r,p);
113         break;
114     else
115         disp('Error, kun 1 og 2 er mulig')
116     end
```

```
117 end
118 end
```

8.2.18 Funksjon: makeConfusion.m

```
function [tAcc,PPV,NPV,Sen,Spe,tAccMat,PPVMat,NPVMat,SenMat,SpeMat,...
    results] = makeConfusion(result,visVerdier,mappe)
% Denne funksjonen slaar sammen alle forvirringsmatrisene i cellestrukturen
% result til en cellestruktur. Gir ogsaa ut andre ytelsesmaal
% Inn:
% Result    = cellestruktur med forvirringsmatriser av storrelse rxp hvor r
% er antall opptak og p er antall personer
% visVerdier = [1] print tAcc, PPV, NPV osv i command window, [0] ikke
% mappe     = mappe dataene er lagret i
% Ut:
% tAcc      = total noyaktighet
% PPV       = Positiv Prediktiv Verdi
% NPV       = Negativ Prediktiv Verdi
% Sen       = Sensitivitet
% Spe       = Spesifitet
% result    = resultater slaatt sammen til en forvirringsmatrise
% Resten er de samme bare i cellestruktur for hvert opptak og person

if ¬nargin
    [data_dir,¬] = dir_master();
    load(strcat(data_dir,'FunksjonsLoad\','makeConfusionVar'))

end

if strcmp(mappe,'Laerdal')
    recordNums = 1:14;
    personNums = 1:8;
elseif strcmp(mappe,'Opptak05')
    recordNums = 1:8;
    personNums = 1:5;
elseif strcmp(mappe,'Opptak06')
    recordNums = 1:6;
    personNums = 1:5;
end


```

```
37 % TP = result(1,1);
38 % FP = result(2,1);
39 % FN = result(1,2);
40 % TN = result(2,2);
41
42 % Forvirringsmatrise:
43 %   TP  FN
44 %   FP  TN
45
46
47 % TP = true positive
48 % FP = falsk positive
49 % FN = falsk negative
50 % TN = true negative
51
52
53 results = zeros(2);
54 if iscell(result)
55     for r = recordNums
56         for p = personNums
57             results = results + result{r,p};
58             TP(r,p) = result{r,p}(1,1);
59             FP(r,p) = result{r,p}(2,1);
60             TN(r,p) = result{r,p}(2,2);
61             FN(r,p) = result{r,p}(1,2);
62         end
63     end
64
65     tAccMat = (TP+TN)./(TP+TN+FP+FN);
66     tAccMat = extendMat(tAccMat,(TP+TN),(FP+FN));
67
68
69     PPVMat = TP./(TP+FP);
70     PPVMat = extendMat(PPVMat,TP,FP);
71
72     NPVMat = TN./(TN+FN);
73     NPVMat = extendMat(NPVMat,TN,FN);
74
75     SenMat = TP./(TP+FN);
76     SenMat = extendMat(SenMat,TP,FN);
```

```
77
78      SpeMat = TN./(TN+FP);
79      SpeMat = extendMat(SpeMat,TN,FP);
80  else
81      results = result;
82      tAccMat = [0 0; 0 0];
83      PPVMat = [0 0; 0 0];
84      NPVMat = [0 0; 0 0];
85      SenMat = [0 0; 0 0];
86      SpeMat = [0 0; 0 0];
87  end
88
89
90
91
92  % Wikipedia¬Confusion_matrix
93  tAcc = (results(1,1)+results(2,2))/sum(sum(results));
94  PPV = results(1,1)/sum(results(:,1));
95  NPV = results(2,2)/sum(results(:,2));
96  Sen = results(1,1)/sum(results(1,:));
97  Spe = results(2,2)/sum(results(2,:));
98
99
100
101
102
103
104 if visVerdier
105     results
106     disp(['Total noyaktighet:       ' num2str(tAcc)])
107     disp(['Positiv prediktiv verdi: ' num2str(PPV)])
108     disp(['Negativ prediktiv verdi: ' num2str(NPV)])
109     disp(['Sensitivitet:            ' num2str(Sen)])
110     disp(['Spesifitet:              ' num2str(Spe)])
111 end
112 end
113
114 function XXXmat = extendMat(XXXmat,T1,T2)
115 r = size(XXXmat,1);
116 p = size(XXXmat,2);
```

```
117
118 XXXmat(r+1,:) = sum(T1,1)./(sum(T1,1)+sum(T2,1));
119 XXXmat(:,p+1) = [sum(T1,2)./(sum(T1,2)+sum(T2,2));sum(sum(T1))...
120     ./(sum(sum(T1))+sum(sum(T2)))];
121
122
123 end
```

8.2.19 Funksjon: saveFigs.m

```
function [] = saveFigs(name,figHandle,saveplace,forceOverWrite,posisjon)
%Bruk denne funksjonen til aa lagre figurer
% Inn:
% name          = filnavn til figur som skal lagres
% figHandle     = figurhandle, gcf vanligvis
% saveplace     = lagringsplass, [1-5]
% forceOverWrite= [1] lagre over eksisterende fil uansett, eller ikke [0]
% posisjon      = posisjon til figur, storrelse eks:[100, 100, 1000, 800]

[¬,img_dir,img_rapport,img_gjort] = dir_master();

if nargin == 0
    figHandle = gcf;
    name = input('Type in figure name: ','s');
    saveplace = input(['Type in place to save figure: [1]img_dir, '...
        '[2]img_rapport, [3]img_gjort, [4]mote_dir, [5]fig_temp: ']);
    forceOverWrite = 0;
elseif nargin == 1
    figHandle = gcf;
    saveplace = input(['Type in place to save figure: [1]img_dir, '...
        '[2]img_rapport, [3]img_gjort, [4]mote_dir, [5]fig_temp: ']);
    forceOverWrite = 0;
elseif nargin == 2
    saveplace = input(['Type in place to save figure: [1]img_dir, '...
        '[2]img_rapport, [3]img_gjort, [4]mote_dir, [5]fig_temp: ']);
    forceOverWrite = 0;
elseif nargin == 3
    forceOverWrite = 0;
end
mote_dir ='C:\Users\oyvin\Dropbox\Masteroppgave\Matlab\Figur mote\170425\';

while true
    switch saveplace
        case 1
            savedir = strcat(img_dir,name);
        case 2
```

```
37              savedir = strcat(img_rapport,name);
38          case 3
39              savedir = strcat(img_gjort,name);
40          case 4
41              savedir = strcat(mote_dir,name);
42          case 5
43              n = 1;
44              name_ = name;
45              while 1
46                  name = strcat(name_,'_',num2str(n));
47                  savedir = strcat(img_dir,'TempFiles\',name);
48                  if exist(strcat(savedir,'.png'))
49                      n = n+1;
50                  else
51                      break;
52                  end
53              end
54
55          otherwise
56              savedir = strcat(img_rapport,name);
57      end
58      if exist(strcat(savedir,'.png'),'file')&&¬forceOverWrite
59          disp(strcat(savedir,'png'))
60          choose = input(['A file with same name exist in this directory.'...
61              ' Do you want to overwrite it? [y],[n],[q]: '],'s');
62      else
63          choose = 'y';
64      end
65
66      if strcmp(choose,'y')
67          if nargin<5
68              set(figHandle, 'Position', [100, 100, 1000, 800]);  % Set size
69          elseif nargin ==5
70              set(figHandle, 'Position', posisjon);  % Set size
71          end
72          saveas(figHandle,savedir,'png')%png
73          saveas(figHandle,savedir,'epsc')%epsc
74  %         saveas(figHandle,savedir,'pdf')%epsc
75          if saveplace == 4
76              savefig(figHandle,savedir)
```

```
77          else
78              savefig(figHandle,strcat('F:\Backup\Masteroppgave\Figdata\',...
79                  name))
80          end
81          disp('File was saved')
82          break;
83      elseif strcmp(choose,'n')
84          name = input('Type in figure name: ','s');
85      elseif strcmp(choose,'q')
86          disp('File was not saved')
87          break;
88      end
89  end
90
91  end
```

8.2.20 Funksjon: tabGen.m

```
function [] = tabGen(tab,topTitle,leftTitle,dec,caption,label,pixels,tab2)
%Funksjon for a generere Latex tabeller fra en matrise eller cellestruktur
% Inn:
% tab       = hovedtabell
% topTitle  = tittel pa toppen
% leftTitle = tittel til venstre
% dec       = antall gjeldende tall pa data i tabellen
% caption   = caption til figuren
% label     = label til figuren
% pixels    = pixels mellom text og kolonne default = 6
% tab2      = annen tabell, optional [tab/tab2]

if ¬nargin % Brukt for testing
    clc
    load(strcat(data_dir,'FunksjonsLoad\','tabGenVar'))
    tab = tAccMatrix;
    dec = 2;
end

% Tabell 2 er valgfri
if nargin == 7
    tab2{size(tab,1),size(tab,2)} = [];
end


% Far begge tabellene over pa celleformat
if ¬iscell(tab)
    if ismatrix(tab)
        tab = num2cell(tab);
    end
end

if ¬iscell(tab2)
    if ismatrix(tab2)
        tab2 = num2cell(tab2);
    end
```

```
37 end
38
39
40 % Far begge topTitle over pa celleformat
41
42 if ¬iscell(topTitle)
43     if ismatrix(topTitle)
44         topTitle = num2cell(topTitle);
45     end
46 end
47
48 % Far begge leftTitle over pa celleformat
49 if ¬iscell(leftTitle)
50     if ismatrix(leftTitle)
51         leftTitle = num2cell(leftTitle);
52     end
53 end
54
55 % Slar sammen begge tabeller, topTitle og leftTitle
56 tabTotal(2:length(leftTitle)+1,1)  = leftTitle;
57
58 tabTotal(2:size(tab,1)+1,2:size(tab,2)+1) = tab;
59
60 tabTotal(1,1:size(topTitle,2)) = topTitle;
61
62 tab2_ = tab2;
63 clear tab2
64 tab2(2:size(tab,1)+1,2:size(tab,2)+1) = tab2_;
65
66
67 R = size(tabTotal,1);   % Rader
68 C = size(tabTotal,2);   % Kolonner
69
70
71 % Skriver gradvis ut mer og mer i command window
72 str = [];
73 fprintf('\n')
74 fprintf('\n')
75 fprintf('\n')
76 disp('\begin{table}[H]')
```

```
77 disp(['\setlength\tabcolsep{' num2str(pixels) 'pt}'])
78 disp('\centering')
79
80 str = '\begin{tabular}{|';
81
82 for i = 1:size(tabTotal,2)
83     str = [str 'c|'];
84 end
85 str = [str '}'];
86 disp(str)
87 disp('\hline')
88 str = '';
89 % str = [str '\n'];
90
91 for r = 1:R
92
93     for c = 1:C
94
95         if iscell(tabTotal{r,c}) % Kan lage tabell inni tabell
96             R2 = size(tabTotal{r,c},1);
97             C2 = size(tabTotal{r,c},2);
98
99             disp([str '\begin{tabular}{c|c|c|c|}'])
100            disp('\multicolumn{2}{r}{}')
101            disp('&\multicolumn{2}{c}{Prediktert}\\\cline{3-4}')
102            disp(['\multicolumn{1}{r}{\multirow{3}{*}{\rotatebox{90}'...
103            '{Virkelig}}}&\multicolumn{1}{r|}{}&P&N\\\cline{2-4}'])
104            str = '';
105
106
107            leftName = {'P','N'};
108            for r2 = 1:R2
109                str = [str '&' leftName{r2} '&'];
110                for c2 = 1:C2
111                    str = [str num2str(tabTotal{r,c}{r2,c2},6)];
112                    if c2<C2
113                        str = [str,'&'];
114                    else
115                        str = [str,'\\\cline{2-4}'];
116                        disp(str)
```

```
117                         str = '';
118                     end
119                 end
120             end
121             disp('\end{tabular}')
122 
123         else
124             if nargin == 8&&r>1&&c>1
125                 str = [str,num2str(tabTotal{r,c},dec),'/',...
126                     num2str(tab2{r,c},dec)];
127             else
128                 str = [str,num2str(tabTotal{r,c},dec)];
129             end
130         end
131         if c<C
132             str = [str,'&'];
133         else
134             str = [str,'\\\hline'];
135             disp(str)
136             str = '';
137         end
138 
139     end
140 end
141 fprintf(str)
142 disp('\end{tabular}')
143 disp(['\caption{' caption '}'])
144 disp(['\label{' label '}'])
145 disp('\end{table}')
146 end
```

Figurer

Tabeller