NTNU
Det skapende universitet

BioPatRec som forskningsplatform med fokus på intensjonsestimering og systemtrening

Alf Bjørnar Bertnum

Master i teknisk kybernetikk
Innlevert: juni 2013
Hovedveileder: Øyvind Stavdahl, ITK
Medveileder: Anders Fougner, ITK

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

NTNU
Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

Masteroppgave

Studentens navn: Alf Bjørnar Bertnum

Fag: Teknisk kybernetikk

Tittel (norsk): BioPatRec som forskningsplattform med fokus på intensjonsestimering og systemtrening

Title (english): BioPatRec as a research platform focusing on intent interpretation and system training

Beskrivelse:

BioPatRec er et fremvoksende åpent kildekode-prosjekt, og tilbyr en forskningsplattform for testing og utvikling av algoritmer til protesestyring. Vi vil undersøke om denne plattformen er egnet for våre forskningsområder. Oppgaven omfatter følgende punkter:

1. Utvikle en funksjonsspesifikasjon for et program til en proteseforskningsplattform. Spesiell vekt skal legges på intensjonsestimering og systemtrening for av/på-klassifisering og multifunksjonell proporsjonalstyring.
2. Relater BioPatRec-systemet til de etablerte kravene, og gi en vurdering om hvorvidt det bør innføres som plattform i vår lab.
3. Basert på resultatene i punkt 2, adapter BioPatRec-systemet for bruk i vår lokale lab.

Faglærer: Øyvind Stavdahl, Institutt for teknisk kybernetikk
Veileder: Anders Fougner, Institutt for teknisk kybernetikk

Trondheim, 07.01.2013

Sammendrag

I denne masteroppgaven er intensjonsestimeringen for simultan proporsjonalstyring, utviklet ved institutt for teknisk kybernetikk (ITK) ved Norges teknisk-naturvitenskapelige universitet (NTNU), implementert inn i det internasjonale prosjektet BioPatRec. BioPatRec er et åpent kildekode-prosjekt for proteseforskning, hvor hensikten er å skape en felles forskningsplattform for testing og utvikling av algoritmer for protesestyring.

Inn i implementasjonen av simultan proporsjonalstyring, er det tre styrende faktorer: lineær mapping, ulineær støydemping og justering av forsterkning og terskelverdier. Hver av disse faktorene, i gitt rekkefølge, er selve essensen i intensjonsestimeringen for simultan proporsjonalstyring.

Etter programvarevurderingen konkluderes det at implementasjonen av PGT som systemtrening, bør skje i samarbeid med output-implementasjonen. Fordi disse delene er tett oppknyttet til hverandre, er dette et tema for fremtidig arbeid.

For å unngå språkforvirring, og opprettelsen av for mange nye ord om samme sak, er det foreslått en norsk terminologi for dette forskningsområdet. Forslagene er basert på forskning gjort i engelske artikler, så de er forsøkt navngitt slik at de skal være selvforklarende, men kort beskrevet for å unngå tvil.

Abstract

In this thesis, the intent interpretation for simultaneous proportional control, developed at the Department of Engineering Cybernetics (ITK) at NTNU, is implemented within the international project BioPatRec. BioPatRec is an open-source project for prosthesis development, whereas the goal is to submit a public research platform for testing and development of algorithms for prosthetic control.

Within this implementation of simultaneous proportional control, there are three controlling factors: linear mapping, nonlinear flutter rejection filter and gain and threshold adjustments. Each of these factors, in the given order, is the main essence in the intent interpretation for simultaneous proportional control.

After the evaluation of the software, it is concluded that the implementation of PGT as system training, should be implemented in collaboration with the output implementation. Because of this conclusion, this is a topic for future work.

To avoid any confusion regarding terminology, and the creation of more new additional terms for the same subjects, it is suggested a norwegian termionolgy for this area of research. The suggestions are based on earlier research in english articles, so they're named in the aim of being self-explanatory, but shortly described to avoid any doubt.

Forord

Etter fem år på NTNU, er denne masteroppgaven resultatet av min mastergrad ved teknisk kybernetikk. Det har vært mye utfordringer underveis i oppgaven, derav å sette seg inn i et åpent kildekode-prosjekt, utforske dens virkemåte og finne ut av dens funksjonalitet. Bare det å tilpasse seg et annet prosjekt har vært utfordrende, særlig med tanke på at fokuset bak denne oppgaven har vært modularitet.

Mine takk i denne masteroppgaven går spesielt til Anders Fougner, min veileder og protese-guru i dette forskningsområdet, som alltid har svaret på et problem. Ellers vil jeg takke Øyvind Stavdahl for å ha fortsatt som min faglærer, kommet med innspill der det trengs og godkjent meg til denne oppgaven.

Jeg vil også takke familie og venner for å ha støttet meg underveis i oppgaven. Takk til min medstudent Kristian Berrum, som har vært tett knyttet opp til denne oppgaven, bare med et annet fokusområde, og vært samarbeidspartner under dette prosjektet. Det har vært mange diskusjoner, og vi har begge etter hvert kommet frem til en løsning.

Innhold

Figurer

Tabeller

Nomenklatur

ADC	Analog-til-digital-omformer
DAQ	Datainnsamling (data acquisition)
EMG	Elektromyografi
ITK	Institutt for teknisk kybernetikk
MC	Motion Control
NI	National Instruments
NTNU	Norges teknisk-naturvitenskapelige universitet
PGT	Proteseguidet-trening
SEMG	Overflate-EMG (Surface-EMG)
SGT	Skjermguidet-trening
VRE	Virtual Reality Environment

Del I

Introduksjon

1 Motivasjon

De siste årene har programvaren på ITK basert seg på et LabView-program. Dette programmet har gått i arv fra prosjekt- og masteroppgave over til neste prosjekt- og masteroppgave, i flere omganger. Omsider har dette ført til en litt stor implementasjon, som kan føles til dels kaotisk og tidkrevende å sette seg inn i. Dette er erfaringer fra tidligere arbeid (Bertnum, 2012).

Selv om tidligere prosjekt har vært mer tidkrevende, er det en viktig innsikt fordi en vet hva som bør unngås og forbedres i et mer optimalt program. Det viktige er i alle fall å ha en god mal for et nytt prosjekt, enten om en starter fra bunnen av, om en optimaliserer det nåværende, eller om en kaster seg på et nytt prosjekt - altså et som allerede er påbegynt.

I nyere tid har det blitt publisert et internasjonalt prosjekt for proteseforskning kalt BioPatRec, se Ortiz-Catalan et al. (2013). Hensikten bak dette prosjektet er å lage en felles forskningsplattform for å teste og utvikle algoritmer til protesestyring.

Potensialet bak et internasjonalt prosjekt er at en får et bibliotek med algoritmer til denne typen forskning. En kan også benytte seg av testresultater gjort av andre personer. Et mulig scenario er at en instans har store muligheter og ressurser for å drive testing av proteser, og dermed innhenting av data. Siden testdataen kan deles, kan en like gjerne sitte en annen plass i verden og forske videre på denne dataen. Dette gir mye større fleksibilitet i forbindelse med proteseforskning. Proteser er også dyre investeringer, så en kan sende behandlet data videre til en annen instans, som igjen tester den behandlede dataen opp mot deres proteser. Terskelen for slik forskning blir derfor mye mindre.

I denne masteroppgaven er fokuset på muligheten til å flytte og implementere funksjonaliteten fra ITK-programvaren over i BioPatRec. Siden dette i seg selv kan være en stor oppgave, er hovedfokusert sentralisert rundt selve *intensjonsestimeringen* og *systemtreningen* ved protesestyring. Begge er begreper oversatt fra arbeidet av Fougner et al. (2012). Hvordan denne masteroppgaven er strukturert, presenteres i neste seksjon, under disposisjon.

2 Disposisjon og bidrag

2.1 Disposisjon av masteroppgaven

Del I Introduksjon inneholder motivasjonen, disposisjon og bidrag, samt bakgrunn og mål. Bakgrunnen inkluderer en introduksjon av BioPatRec og ITK-programvaren. Her presenteres ting litt overfladisk, men med litt større fokus på de områdene som er sentrale for denne oppgaven. I tillegg nevnes det litt om mulige pådragsorganer og klasseinndelingen av protesestyringsproblemet. Avslutningsvis er målet med masteroppgaven, hvor det redegjøres for målene og leseren har større innsikt i temaet etter å ha lest bakgrunnen.

Del II Metode beskriver fremgangsmåten brukt i oppgaven. Først presenteres angrepsmetoden brukt for å angripe problemene i oppgaven. Deretter følger en stegvis prosedyre av målene i masteroppgaven, og den er delt inn i tre punkter: funksjonsspesifikasjon, programvarevurdering og implementasjon.

Del III Resultater tydeliggjør hva som er gjort ved å presentere nye implementasjonen i korte trekk, samt oppsummere den norske terminologi og endringer gjort på eksisterende funksjoner i BioPatRec.

Del IV Diskusjon og konklusjon drøfter sterke og svake sider ved fremgangsmåten og resultatene som er oppnådd. Her tas opp de problemstillingene som oppsto underveis, og kritisk analyseres for å forstå roten av de. Basert på diskusjonen og resten av oppgaven, trekkes en konklusjon, og temaer for videre arbeid.

2.2 Liste av bidrag

Funksjonsspesifikasjon Denne spesifikasjonen står i tabell 2 under seksjon 5.2, og inneholder kravene som stilles til BioPatRec.

Norsk terminologi I tabell 4, under seksjon 8, er det gitt et forslag til en standard av begreper for norsk terminologi.

Nye implementasjoner i BioPatRec er oppsummert i tabell 5, under seksjon 9, og står beskrevet kort i samme seksjon. Den detaljerte beskrivelsen står under implementasjonen i seksjon 7 i del II. Koden er å finne i Appendiks A.

Endringer av eksisterende funksjoner i BioPatRec er oppsummert i tabell 6, under del III, og består av små endringer for å legge til ny funksjonalitet. Koden er å finne i Appendiks A.

Forslag til fremtidig arbeid er gitt i seksjon 12, og er ideer å ta tak i ved videre arbeid på dette forskningsområdet.

Web-dokumentasjonen på wiki-sidene (Ortiz-Catalan, n.d.) som er skrevet i denne masteroppgaven, ligger vedlagt i Appendiks B. Denne dokumentasjonen supplerer koden som er implementert. Web-dokumentasjonen (Bertnum, 2013, BioPatRec web documentation) er et arbeid i utvikling, altså vil det mest sannsynlig endres i videre arbeid. Derfor er vedlegget bare et foreløpig utkast.

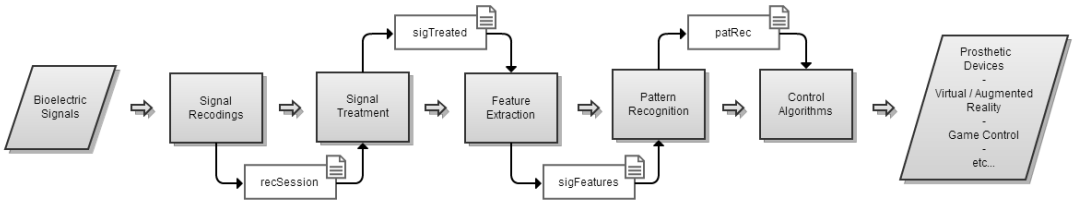
3 Bakgrunn

I denne masteroppgaven skal et åpent kildekode-prosjekt kalt *BioPatRec* analyseres og vurderes opp mot det nåværende programvare-biblioteket hos ITK. Hvis programvurderingen tilsier at systemet er egnet, kan det påbegynnes en implementasjon. For å kunne foreta denne analysen trengs det en oversikt over selve prosjektet, samt en forståelse for prosjektets oppbygning. Derfor vil det gis en introduksjon i programvaren av BioPatRec og ITK. I tillegg beskrives pådragsorganene som er tilgjengelig ved ITK-laben, samt klasseinndelingen av protesestyringsproblemet.

3.1 BioPatRec

Målet for BioPatRec er å skape en felles forskningsplattform for proteseforskning. Ved å tilby dette prosjektet til alle som ønsker å bidra til denne typen forskning, kan enhver person enkelt komme med bidrag. Videre testing og utvikling av algoritmer for protesestyring, samles til en felles plattform. Dermed kan ulike teknikker enkelt sammenlignes og optimaliseres.

Strukturen for dette kapitlet kan illustreres ved hjelp av figur 1, hvor hver modul representer et underkapittel og vil bli forklart mer i detalj.



Figur 1: Modellen illustrerer hvordan veien går i BioPatRec-systemet, altså rekkefølgen på modulene. Dette bildet er hentet fra Ortiz-Catalan (n.d.), og er lisensert under en CC BY-SA 3.0 lisens.

3.1.1 "Signal Recordings"

Inn til denne delen i BioPatRec-systemet kommer de bioelektriske signalene, som blir input-verdiene til systemmodellen. Slike signaler vil ofte være elektromyografi-signaler (EMG-signaler), og kan måles ved hjelp av ulike EMG-sensorer. Et eksempel på en slik sensor er overflate-EMG-sensoren (SEMG-sensoren), som måler EMG-signaler på overflaten av huden. For mer

informasjon om EMG og SEMG henvendes leseren til Bertnum (2012) hvor det står litt mer informasjon.

De bioelektriske dataene målt ved hjelp av sensorer sendes via en eller annen overførings-enhet. Overføringsmetoden brukt i dette prosjektet er å sende dataene ved å bruke en "data acquisition"-enhet (DAQ) levert fra National Instruments (NI). Jobben til en DAQ-enhet vil være å konvertere de analoge signalene fra de bioelektriske målingene om til digitale signaler slik at de lettere kan prosesseres og behandles. En slik omformer kalles en analog-til-digital-omformer (ADC).

Alle målingene gjort av brukeren i *Signal recordings*, gjøres ved at han eller hun observerer en illustrasjon på en skjerm, for deretter og gjengi tolkningen av denne bevegelsen. Med tolkning menes det at brukeren benytter de musklene han selv relaterer til bevegelsen. Denne typen systemtrening kalles skjermguidet-trening (SGT) og er en av flere metoder som kan benyttes til systemtrening.

Prinsippet bak SGT er å presentere en illustrasjon av en bevegelse i form av enten et bilde, animasjon eller video, for deretter å gjengi eller følge denne bevegelsen. SGT beskrives mer i seksjon 3.2.4. Ved å fullføre denne treningen genereres et sett med brukerdata nødvendig for den videre databehandlingen i systemet.

3.1.2 "Signal Treatment"

Etter å ha målt signalet er det nødvendig å behandle det før en beregner utvalgte egenskaper. Her er det flere steg som spiller inn, og de implementerte rutinene er dokumentert hos Ortiz-Catalan (n.d.).

Et steg i behandlingen av signalet er å fjerne uønskede bevegelser og irrelevante kanaler. Eksempel på irrelevante kanaler er målinger der det i utgangspunktet ikke burde være noe utslag på målingene. Dette betyr deteksjon av muskelkontraksjoner, hvis EMG blir målt, i kanaler som ikke burde være involvert i denne bevegelsen.

Dersom det blir oppdaget transiente perioder i muskelkontraksjoner, bør disse fjernes. Slik unngås potensiell ustabilitet av de variasjonene som gis i de høye og lave amplitudene av målingene.

Ut i fra målingene vil det være mulig å definere en hvile-tilstand, altså en tilstand med ingen bevegelse. Dette skjer automatisk basert på måledata, for slik å tilpasse tilstandene til målingene.

Filtrering av signalet vil være et viktig steg for å redusere støy. På den

måten fremheves det egentlige signalet og støy vil ha en dempet forplantning videre i prosesseringen, hvilket er ønskelig. Foreløpig er det diverse frekvens- og rom/*spatial*-filtre implementert i systemet, men med mulighet for legge til flere. Det er også mulighet for å segmentere datavinduene til overlappede- eller ikke-overlappede-vinduer, hvorav en kan justere størrelsen til trenings-, valideringss- og testings-settene.

3.1.3 "Feature Extraction"

De fleste BioPatRec algoritmene krever diskrete input-signaler, men fra *Signal Treatment* vil signalene være i tid- eller frekvens-domenet. Altså når eventuell behandling av signalet er gjort, må en beregne de utvalgte signalegenskapene for å oppnå diskrete data. Noen av de tilgjengelige egenskapene står listet opp i tabell 1, som er en tabell hentet fra Ortiz-Catalan (n.d.).

3.1.4 "Pattern recognition"

I selve mønstergjenkjenningen er modulen delt i to; offline og sanntid. Offline-delen er designet slik at den baserer seg på forhåndsregistrerte data til å trene opp en algoritme. Slik kan systemet trenes opp, valideres og testes. Dette skjer uavhengig av om systemet er koblet opp mot en faktisk enhet, som for eksempel en protese som mottar bevegelses-instrukser. Basert på data registrert i sanntid, bruker sanntids-delen den opptrente algoritmen til å beregne pådrag til pådragsorganet i sanntid.

Ved å benytte seg av offline-delene, skjer en algoritmetrening som skaper en relasjon mellom input-signalene i systemmodellen og output-signalene som går til protesen eller andre pådragsorganer. Med denne relasjonen menes det å skape en kobling mellom de bioelektriske signalene som kommer inn i modellen og de eventuelle bevegelsesklassene, se tabell 4 i seksjon 8, som representerer de ulike pådragene til pådragsorganene. For eksempel vil en skape en relasjon slik at aktivitet i *m. pronator teres* skal assosieres med pronasjon i underarmen. Dette er naturlig i kroppslig sammenheng siden denne relasjonen allerede er til stede, men for en protese må denne koblingen skapes.

I tillegg til å trene algoritmene i offline-delen, skjer det en testing av den trenede algoritmen. Formålet bak dette er å gi en prediksjon på hvilken bevegelse et gitt datasett vil gi, samt sannsynlighetene for at det gir de andre bevegelsene. Således får en et inntrykk av hvordan denne algoritmen vil prestere, og om målingene i opptaket ga fornuftige resultater. I tilfeller hvor sensorer er dårlig plassert, kan dette gi et pek på om oppførselen til algoritmen er optimal.

ID	Feature	Domain
tmn	Mean	Time
tmabs	Mean Absolute value	Time
tmd	Median	Time
tstd	Standard deviation	Time
tvar	Variance	Time
twl	Waveform length	Time
trms	RMS	Time
tzc	Zero-crossing	Time
tpks	Peaks (over RMS)	Time
tmpks	Peaks mean	Time
tmvel	Mean velocity	Time
tslpch1	Slope changes (peaks)	Time
tslpch2	Slope changes (diff)	Time
tpwr	Power	Time
tdam	Difference abs. mean	Time
tmfl	Max fractal length	Time
tfd	Fractal dimension	Time
tfdh	Fractal dim. Higuchi	Time
tren	Rough entropy	Time
tcr	Correlation	Time
tcv	Co-variance	Time
fwl	Waveform length	Frequency
fmn	Mean	Frequency
fmd	Median	Frequency
fpmn	Peaks mean (Top 5)	Frequency
fpmd	Peaks median (Top 5)	Frequency
fpstd	Peaks std (Top 5)	Frequency

Tabell 1: Tabell med noen av signalegenskapene tilgjengelig i BioPatRec-systemet. Tabellen er hentet fra Ortiz-Catalan (n.d.).

Sanntids-delen av programmet har som funksjon å kjøre protesestyring i sanntid. I kjøringen vil dette skje ved at bioelektriske data måles, og deretter assosieres med én eller flere klasser i sanntid. For at dette skal være mulig, benytter sanntids-delen seg av en algoritme trent opp under offline-modus. Slik vil algoritmen ha de nødvendige opplysningene og parametrene for å kunne kjøre algoritmen i sanntid. Siden dette kan ta litt tid er det nødvendig å gjøre dette i forkant av kjøringen i sanntid, derav en egen offline-del.

3.1.5 "Control Algorithms"

Styringsalgoritme-modulen har som funksjon å delegere signalene gitt av mønstergjenkjennings-algoritmene i forrige modul, se seksjon 3.1.4. Deretter fordeles signalene på pådragsorganene slik at en styring oppnås. En kan se på det slik som at styringsalgoritmen estimerer en optimal koblingen mellom input-signalene og pådraget som skal påsettes pådragsorganet. Pådragsorganet kan være være proteser, *Virtual Reality Environment* (VRE), spillstyring eller andre enheter som ønskes å styre ved hjelp av bioelektriske signaler.

Hvis ikke det benyttes en styringsalgoritme, må koblingene mellom mønstergjenkjenningen og output-delen hardkodes inn i programmet. Dette er en lite optimal måte å estimere pådraget på, da pådragene ikke gjennomgår noen form for post-prosessering. Poenget med post-prosesseringen er å forbedre stabiliteten rundt sanntidskjøringen til systemet (Ortiz-Catalan et al., 2013); da i form av å oftere velge rett pådrag til pådragsorganet, og se bort fra sporadiske feil i pådragsestimatene. Feil i pådragsestimatene kan ses på som en feilklassifisering.

3.2 ITK-programvaren

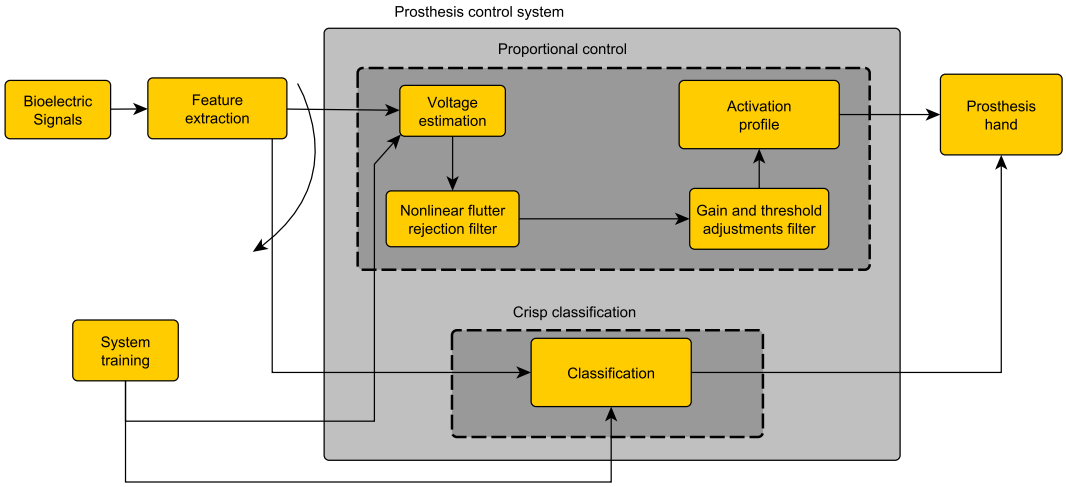
Dette kapitlet vil basere seg på arbeidet gjort i Bertnum (2012), og vil i stor grad være en omskriving med modifikasjoner av dette arbeidet.

Strukturen på denne seksjonen vil basere seg på en adaptert systemmodell fra arbeidet av Bertnum (2012), se figur 2. Denne modellen illustrerer dataflyten fra input til output i ITK-programvaren. Slik får en lett oversikt over hvilke metoder som blir benyttet i programmet og hvor dataen blir behandlet. Modellen er relativt lik den forrige av design og tankegang, men noen detaljer er lagt til og noe er forandret.

Beskrivelsen av modellen vil ta utgangspunkt i det som står i tidligere arbeid (Bertnum, 2012), men med en del omskriving og andre detaljer. Overskriftene i følgende kapitler vil ta for seg stegene i figur 2, men med norske titler. Titlene er relativt rett frem, med unntak av *Voltage estimation* som i teorien er en *lineær mapping*.

3.2.1 Beregning av signalegenskaper

Denne bolken tar for seg flere operasjoner i ITK-programvaren kontra det som skjer i BioPatRec-systemet. I dette tilfellet betyr det at både behandlingen av de bioelektriske signalene, og beregningen av signalegenskaper, vil skje i den



Figur 2: Modell av protesesytemet i ITK-programvaren. Her illustreres flyten fra input til output i programmet, og det er to alternative styringssystemer for protesestyring; proporsjonalstyring(“*Proportional control*”) og av/på-klassifisering (“*Crisp classification*”). Denne figuren er adaptert fra Bertnum (2012).

samme bolken. En alternativ måte å observere behandlingen av data, er at det skjer i overgangen fra blokken *Bioelectric signals* til *Feature extraction*.

Beregningen av signalegenskaper er svært lik både for ITKs nåværende programvare og det nye BioPatRec-systemet. Begge gjør et opptak av de bioelektriske-signalene for deretter å behandle og formatere de etter angitte metoder. I motsetning til BioPatRec-systemet, vil ikke ITK-programvaren tilby muligheten til å spesifisere formateringsmetoder for de målte signaldataene i det grafiske grensesnittet.

3.2.2 Proporsjonalstyring

Det er to mulige styringssystemer for protesestyring i ITK-programvaren; proporsjonalstyring, som vil bli beskrevet i dette kapitlet, og av/på-klassifisering, som vil bli beskrevet i neste kapittel (3.2.3).

Proporsjonalstyring er den metoden som er gjort mest forskning på i senere tid på NTNU, og den som vil bli fokusert mest på i denne oppgaven. Den siste publiserte forskningen fra NTNU til nå er fra Fougner (2013). Sett bort ifra det, er denne metoden lite dokumentert. Dette skyldes at mye forskning tar utgangspunkt i andre aktivasjonsprofiler (Fougner et al., 2012). En aktivasjonsprofil baserer seg på hvordan bevegelsesprofilen ser ut

i henhold til aktiveringen i pådragsorganet. Det vanlige er av/på-profil, eller rampefunksjon-profil. For et større overblikk over dette, henvendes leseren til figur 2 i (Fougner et al., 2012, s. 667). Grunnen til disse valgene har i stor grad vært deres grad av robusthet og forutsigbarhet (Fougner et al., 2012, s. 668). Ulempen vil selvfølgelig være at de ikke kan styre pådraget til pådragsorganet på en proporsjonal måte.

Prinsippet bak proporsjonalstyring er å styre en output til en protese med en variabel input. Denne metoden ble introdusert i arbeidet av Fougner et al. (2012). Brukeren kan altså variere styre-inputen innen et kontinuerlig intervall for å styre i alle fall én mekanisk output på protesen. Selv om styre-inputen kan varieres innenfor et kontinuerlig intervall, må den likevel korrespondere med det kontinuerlige intervallet på outputen. Dette gir mening siden intervallet på den mekaniske outputen nødvendigvis vil være den begrensede faktoren for den mekaniske styringen.

Simultan proporsjonalstyring I ITK-programvaren er det implementert støtte for simultan styring ved proporsjonalstyring. Dette betyr at brukeren kan styre flere frihetsgrader på en gang. Slik vil brukeren optimalt sett kunne styre protesen både raskere og mer intuitivt. I arbeidet av Fougner et al. (submitted) kunne en se tegn på at simultan proporsjonalstyring ga lovende resultater, men som påpekt i artikkelen er det ikke konkluderende resultater. Det var behov for et bedre test-oppsett og flere forsøkspersoner for å dra noen endelig konklusjon. Likevel er dette viktig forskning og en nyttig implementasjon i programvaren, og vil være essensiell i forskning fremover.

Lineær mapping. For å skape den relasjonen nevnt tidligere mellom input og output, trengs en form for estimeringsmetode. Disse metodene ble implementert i arbeidet av Linnerud (2012), gitt *lineær mapping* og et system modellert som i ligning 3.1.

$$\vec{F} = \mathbf{A}\vec{x} \tag{3.1}$$

Her representer $\vec{F}$ en vektor som inneholder referanseverdiene som sendes til neste steg i protesestyringssystemet. $\mathbf{A}$ er matrisen som relaterer $\vec{x}$ til $\vec{F}$, og $\vec{x}$ er en vektor av input-data. $\vec{x}$ vil som regel inneholde egenskapene som er beregnet ifra input-signalet.

Lineær regresjon. I ligning 3.1 vil $\vec{F}$ og $\vec{x}$ være kjent etter systemtreningen, mens $\mathbf{A}$ er ukjent. Derfor trengs en estimator for å estimere $\mathbf{A}$. Til dette formålet

kan lineær regresjon være et alternativ. Dette er en metode for å best mulig finne en lineær funksjon hvor resultatet er mest mulig likt utgangspunktet. I arbeidet av Linnerud (2012), ble ligning 3.2 implementert i ITK-programvaren som en *Matlab*-funksjon. En notis er at i selve masteroppgaven til Linnerud (2012), er det en skrivefeil i formlene for lineær regresjon, men dette er implementert rett i programvaren. Dette er altså snakk om en transponering for lite.

$$\mathbf{A} = (\vec{x'} \backslash \vec{F'})' \tag{3.2}$$

Operatorene "\" og "/" blir i *Matlab* tolket avhengig av formen på argumentene i systemmodellen $F = Ax$.

Normal lineær estimering. Det eksisterer to estimeringsmetoder i ITK-programvaren, normal lineær estimering og dekoblet lineær estimering. Førstnevnte tar i bruk lineær regresjon (ligning 3.2) direkte for å estimere **A**. Deretter vil ligning 3.1 brukes for å kjøre proporsjonalstyring i sanntid, hvor da $\vec{F}$ vil være den ukjente. Resultatet vil være de nødvendige output-verdiene for videre beregning i protesestyringssystemet. Dette kan være til for eksempel aktuatorer på et pådragsorgan som ønskes å styre.

Dekoblet lineær estimering Denne delen er i stor grad basert på det beskrevet i arbeidet av Linnerud (2012), og er en upublisert metode utviklet av Øyvind Stavdahl og Anders Fougner. I stedet for å bruke lineær regresjon direkte, så dekomponeres referanseverdiene inn i ulike typer egenskapsvektorer. Egenskapene som trekkes ut er gjennomsnitt, standardavvik og form, slik som i ligning 3.3, 3.4 og 3.5.

$$\vec{F_{mean}} = \vec{mean}(F) \tag{3.3}$$

$$\vec{F_{std}} = \vec{std}(F) \tag{3.4}$$

$$\vec{F_{form}} = \frac{\vec{F} - \vec{F_{mean}}}{\vec{F_{std}}} \tag{3.5}$$

Deretter benyttes lineær regresjon på disse egenskapsvektorene for å estimere **A**. Dette gir formlene 3.6, 3.7 og 3.8.

$$\mathbf{A_{mean}} = (\vec{x'} \backslash \vec{F'_{mean}})' \tag{3.6}$$

$$\mathbf{A_{std}} = (\vec{x'} \backslash \vec{F'_{std}})' \tag{3.7}$$

$$\mathbf{A_{form}} = (\vec{x'} \backslash \vec{F'_{form}})' \tag{3.8}$$

Til slutt vil ligning 3.9 brukes for å beregne $\vec{F}$ i sanntid. Her er $.*$ en *Matlab*-operator for elementvis multiplikasjon.

$$\vec{F} = (\mathbf{A_{form}}\vec{x}) .* (\mathbf{A_{std}}\vec{x}) + \mathbf{A_{mean}}\vec{x} \tag{3.9}$$

Problemet med denne estimatoren er at den ikke gir noen fordeler ovenfor den normal lineære estimatoren. Altså er det ikke noe poeng i å bruke denne metoden for lineære problemer. Hvis derimot en av ligningene for beregningen av $\mathbf{A}$ er ulineære, så vil denne estimatoren være aktuell. Dette er en tanke av Anders Fougner, og er et tema for videre arbeid.

Ulineær støydemping Etter at signalet er blitt lineært avbildet, gjøres det en ulineær støydemping på signalet. Dette er for å dempe raske og små variasjoner i amplituden på input-signalet, slik at protesen holder en konstant hastighet. Filterdesignet kan ses i figur 3 under "Nonlinear flutter rejection filter". Ulineariteten sett i figuren er definert som $y = |x|tanh(kx)$, og er i prinsippet en glatt tilnærming av et dødbånd-filter. Dette er beskrevet i artikkelen av (Fougner et al., submitted, s. 2).

Justering av forsterkning og terskelverdier Hvis målingene fra system- og algoritmetreningen er litt svake, altså at amplituden på treningsdata er litt lav i forhold til referansedata, kan en sette på en forsterkning. Denne forsterkningen vil gjøre slik at brukeren lettere oppnår de høyeste pådragene til pådragsorganet. Dette skjer gjerne mens protesestyringen foregår i sanntid, for da oppdager en raskt om brukeren sliter med å gi på maks pådrag til pådragsorganet. Likevel vil en tidlig kunne se på sammenligningen av treningsdata opp mot referansedata om det trengs å legge på en forsterkning.

I justeringen av forsterkning og terskelverdiene defineres områdene hvor de forskjellige frihetsgradene skal fungere. I praksis vil dette definere hvordan pådragsorganet skal oppføre seg etter som sanntidsmålingene plottes i en figur som den under "Gain and threshold adjustments" i figur 3. Alt innenfor den indre sirkelen, markert i rødt, er definert som hvile-tilstand. Langs x-aksen, område markert i blått, defineres lukking og åpning av hånden, hvorav

førstnevnte er definert for positive verdier og vice versa for åpning av hånden. Det samme gjelder langs y-aksen, område markert i grønt, for pronasjon og supinasjon av underarmen, hvorav førstnevnte også gjelder for positive verdier, og motsatt for supinasjon. Områdene i mellom, altså de hvite områdene, vil representere den simultane styringen av pådragsorganet. Dette står også forklart i artikkelen av (Fougner et al., submitted, s. 4).

Når et pådrag befinner seg i et av operasjonsområdene definert fra forrige avsnitt, vil pådraget som sendes til protesen endres avhengig av området en befinner seg i. Hvis en befinner seg i områdene langs x- eller y-aksen, settes det andre pådraget lik null. Når en er i området for simultan styring, settes det et likt pådrag på begge frihetsgrader. Det resterende området er definert av hvile-terskelen, og er pådraget innenfor denne terskelen, settes alle pådragene lik null.

Aktivasjonsprofil For å generere den aktivasjonsprofilen brukt i ITK-programvaren, se figur 3, brukes to sigmoid funksjoner oppe på hverandre. Slik oppnås en aktivasjonsprofil hvor flest tilfeller, vil gi en høy eller lav fart/kraft på pådragsorganet. Selv om en fortsatt har muligheten til å styre proporsjonalt, har brukeren lettere for å styre på lave verdier for presisjon og høye verdier for grov-arbeid. Dette tyder på at det blir et hybrid system mellom *multi-level*-styring og proporsjonalstyring(Fougner et al., submitted, s. 4). Likevel, som påpekt tidligere, er det fortsatt mulighet for å styre proporsjonalt, altså defineres systemet som proporsjonalt.

Aktivasjonsprofilen ved ITK-programvaren vil fungere som styringsalgoritmen i denne modellen; da den optimaliserer outputen til pådragsorganet. Dette kan ses mer tydelig i figur 3, hvor aktivasjonsprofilen delegerer motorspenning til et pådragsorgan, i dette tilfellet en håndprotese.

3.2.3 Av/på-klassifisering

Av/på-klassifisering er en av metodene som blir benyttet i ITK-programvaren. Denne metoden tar i bruk mønstergjenkjenning for å gjenkjenne et spesifikt av/på-sett. Disse av/på-settene vil definere et spesifisert antall klasser hvor hvert sett er en klasse. Hver klasse består så av tilhørende karakteristiske funksjoner med kodomenet $\{0, 1\}$. Detaljer rundt av/på-klassifisering kan finnes i Rybkin et al. (2005).

Effekten av av/på-klassifisering er at når et signal registreres, gjenkjennes det enten som én eller flere klasser, eller ikke i det hele tatt. De klassene som gjenkjennes vil aktiveres, og de som ikke gjenkjennes blir satt som ikke-aktivert.

I ITK-programvaren er det foreløpig bare implementert støtte for én bevegelse i gangen med av/på-klassifisering. Proporsjonalstyring har derimot mulighet for simultan styring. I tillegg er det også bare én grad av styrke ved av/på-klassifisering, altså må brukeren forholde seg til én styrke på hver bevegelse. Dette krever at brukeren må lære å tidsestimere bevegelsene, altså å vite når en skal starte og stoppe en bevegelse, for å oppnå spesifikke posisjoner med for eksempel en protese.

Problemet med én grad av styrke er hvis brukeren ønsker å gjøre både en hurtig og en rolig bevegelse. Hvis brukeren skal ha mulighet til dette, må det legges til ekstra-funksjonalitet på protesen som tillater dette, for eksempel en av/på-bryter for hurtige eller rolige bevegelser. Dette vil bli en begrensning ved denne typen styring. Til gjengjeld er av/på-klassifisering en robust og forutsigbar form for protesestyring (Fougner et al., 2012).

3.2.4 Systemtrening

Ved å benytte seg av *systemtrening*, et begrep innført av Fougner et al. (2012), kan brukeren trene opp systemet for protesestyring. Dette skjer ved at brukeren spesifikt for egne målinger relaterer input-signalene i systemmodellen med output-signalene som går til protesen eller andre pådragsorganer. Det er to inndelinger av dette i ITK-biblioteket: skjermguidet-trening (SGT) og proteseguidet-trening (PGT). Disse står også beskrevet i arbeidet av Bertnum (2012), som tar utgangspunkt i Fougner et al. (2012).

Brukertrening er et annet begrep i artikkelen av Fougner et al. (2012) som er verdt å nevne. Dette går ut på trening av brukerens evne til å faktisk styre pådragsorganet. Dette aspektet er noe som kommer med trening og brukererfaring. Etter hvert som brukeren styrer pådragsorganet og lærer bedre å bruke musklene som aktiverer de forskjellige bevegelsene, lærer han også å trene protesen samt å styre den bedre. Dette påpekes også i arbeidet av Bertnum (2012), og ble merket allerede i tidlige stadier av protesetrening og -bruk. Dette ordet er lagt til i tabell 4.

SGT Skjermguidet-trening tar utgangspunkt i at en bruker observerer en skjermguidet-instruksjon for å trene opp systemet. Instruksjonen kan være i form av et bilde, en animasjon eller en film. Prinsippet er bare å vise på en skjerm hva brukeren skal prøve å etterligne. Denne metoden er brukt i arbeidet av Fougner (2013).

Problemet med SGT er at brukeren blir avhengig av en skjerm og ekstern maskin til treningen, og det er ikke alltid tilfellet at brukeren er i besittelse

disse. Likevel, som påpekt i Bertnum (2012), er teknologien i rask fremgang; mobile maskiner, for eksempel smart-mobiler, er for flere et vanlig gode. Derfor kan dette være en løsning på eventuell skjerm- og ekstern maskinmangel.

I arbeidet av Chicoine et al. (2012) påpekes en av fordelene med SGT, og det er muligheten for å skreddersy den skjermguidede treningen til brukeren. Uansett er brukeren begrenset til en skjerm, så all kommunikasjon må foregå gjennom den. Derfor kreves det at brukeren har en kognitiv forståelse av det som foregår på skjermen.

PGT Proteseguidet-trening er en metode introdusert i arbeidet av Lock et al. (2011) og Simon et al. (2011). Basert på deres forskningsresultater med mønstergjenkjenning som styringssystem, har denne metoden vist seg som et meget godt alternativ til den ellers vanlige SGT-metoden. Konseptet med metoden er i grunn veldig enkel og intuitiv, protesen beveger seg, og brukeren aper etter med et imaginært lem. Dermed aktiveres de spesifikke musklene som brukeren assosierer med selve bevegelsen i protesen. Brukeren skaper en direkte relasjon mellom egne muskler og protesen, altså skreddersys input-dataen til protesen. På grunn av varierende oppførsel i pådragsorgan, for eksempel treghetsmoment i en protese, vil en skreddersying bidra til en bedre brukeropplevelse av pådragsorganet.

Fordelen med PGT er at en ikke er avhengig av en skjerm og ekstern maskin for å gjøre en systemtrening. Derfor er denne metoden praktisk for en hverdagslig protesebruker. På grunn av dens raske trening, sammenlignet med eldre metoder, og enkelthet er den også brukervennlig. I ITK-programvaren fungerer også denne metoden for proporsjonalstyring. Dette har vist seg som en veldig bra metode for systemtrening med denne typen styringssystem (Fougner et al., submitted). For en detaljert oversikt over øvelsene praktisert med PGT i ITK-programvaren, se Bertnum (2012).

3.3 Pådragsorganer

Etter et potensielt protesesystem har gjort oppgaven med å relatere en pasients muskelbevegelser til ulike pådrag, vil disse signalene sendes til et pådragsorgan. Et eksempel på et slikt pådragsorgan vil være en håndprotese. I prinsippet skal dette pådragsorganet kunne være hva som helst, men det må implementeres oversettingsfunksjoner som kan oversette output-dataene fra styringsalgoritmene til pådragsorganet i form av motor-spenning, eller hva annet pådragsorganet krever.

ITK-programvaren støtter styring av en proteseløsning levert av Motion

Control (MC), som består av en protesehånd og et rotasjonsledd. I tillegg er det støtte for en robothånd laget i forbindelse med prosjekt- og masteroppgaver på NTNU.

MC-protesehånden kan bare åpnes eller lukkes, men i laben på NTNU benyttes denne i kobling med en håndleddsrotator levert av MC, se figur 6. Slik får brukeren et større utvalg av bevegelser, og kan utføre flere håndrelaterte oppgaver.

Robothånden utviklet på NTNU er designet for testing på lab, og for å se protesestyringssystemet i praksis, se figur 7. Hånden har mer funksjonalitet enn MC-protesen, så den vil være mer nyttig hvis en ønsker å eksperimentere med avansert funksjonalitet.

3.4 Klasseinndelingen av protesestyringsproblemet

For lettere å forstå protesestyringsproblemet, har det blitt utarbeidet en modell og klasseinndeling av dette problemet i arbeidet av Fougner et al. (submitted). Inndelingen i figuren er basert på en tidligere modell av Losier, også medforfatter i utarbeidingen av den nye figuren, som representerer de tre hovedstadiene i protesestyring: preprosessering, intensjonsestimering og output. I den nye figuren, se figur 3, kan en se videreutviklingen som er gjort, derav for å inkludere mer av utfordringene rundt protesestyringsproblemet. Til høyre i figuren kan en se et eksempel av et scenario med simultan proporsjonalstyring som sammenlignes med klasseinndelingen.

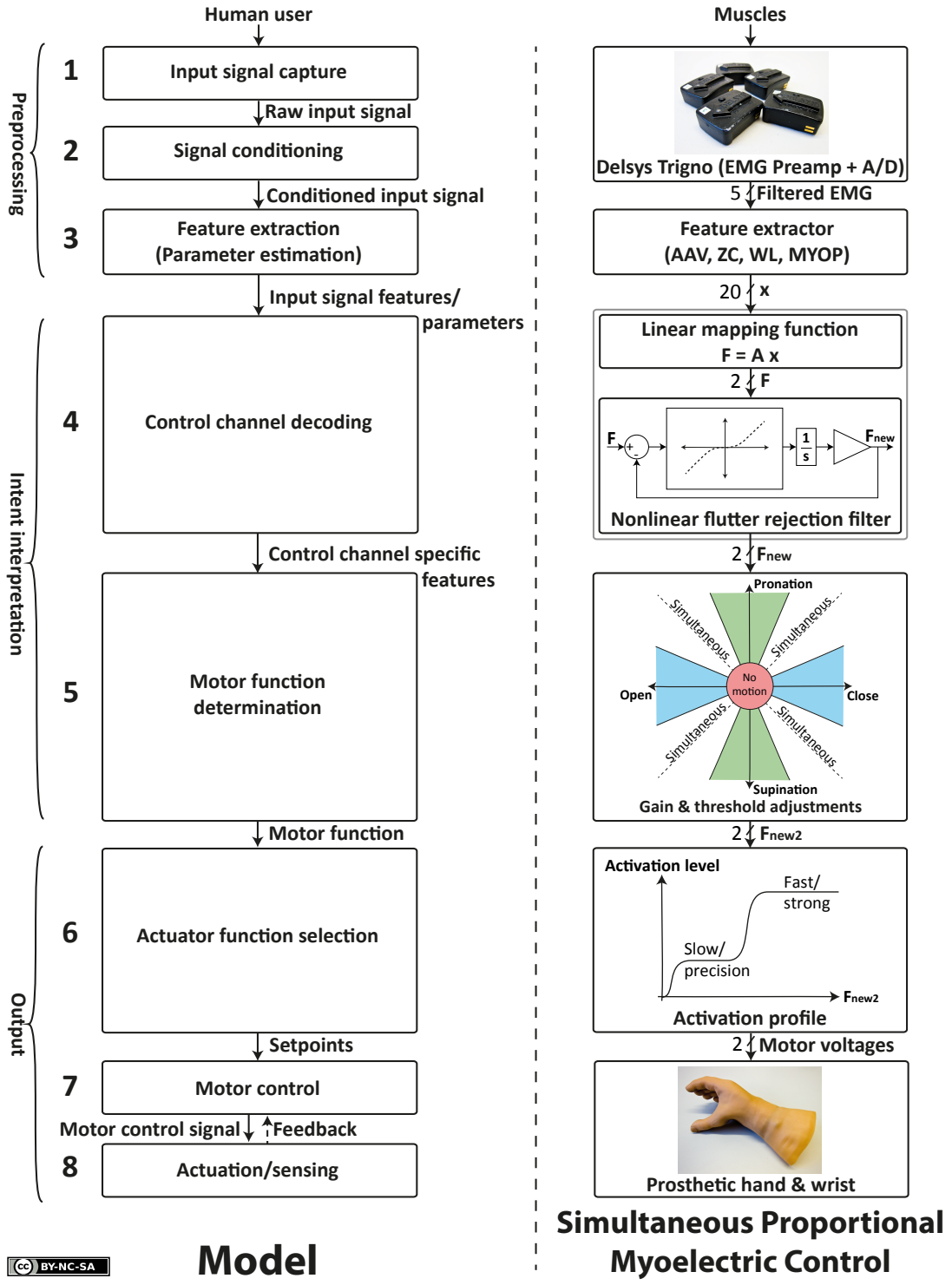
I tillegg til denne modellen, er også systemtrening en viktig faktor. På grunn av at denne ikke direkte inngår i protesestyringen, har den ikke en like naturlig plass i modellen, men i praksis vil den tilføre informasjon til første steg i intensjonsestimeringen. Systemtreningen er den treningen brukeren gjør med protesen, eller pådragsorganet, i forkant av selve protesestyringen. Dette er nødvendig for å kunne gjøre algoritmetreningen, som er den treningen som kreves før en algoritme kan kjøres i sanntid. Dette står mer beskrevet i seksjon 3.2.4.

Første stadiet i modellen er preprosessering, som igjen er delt inn i tre kategorier: input-signal-opptak, signalbehandling og beregning av egenskaper eller parameterestimering. Preprosessering gjør da alt av prosessering på signalet før det skal tolkes. Signalet kommer inn fra brukeren som rådata, altså ubehandlet, for deretter å gjennomgå en behandling, og til slutt en beregning av egenskaper eller parameterestimering.

I intensjonsestimeringen er det to faser, selve dekodingen av styringskanalene og valget av motorfunksjon. Den første fasen har som hensikt å estimere

intensjonen til brukeren ut ifra signalegenskapene i input-signalet, eller ut ifra parametrene. Neste fase er å velge hvilken motorfunksjon estimatet av intensjonen gir, som kan være feil eller rett alt etter som estimatet er optimalt eller ikke.

Siste stadie er output, som er delt inn i tre deler: valget av aktuatorfunksjon, motorstyring og aktivering/sansing. Etter å ha mottatt hvilken motorfunksjoner som skal påsettes ifra del fem, beregner del seks settpunktene for del sju, som blir et valg av pådrag til aktuatorfunksjonene. Settpunktene kan forstås som for eksempel motorspenninger; de forteller hvilken posisjon systemet ønsker protesen skal være i. Deretter skjer motorstyringen, hvor protesen, eller pådragsorganet, mottar pådragene og sender styringssignalene videre til aktiveringen. I del åtte, under aktivering/sansing, skjer selve bevegelsen i protesen, men her er det ofte en kontroller implementert i protesen som gir en tilbakemelding på om protesen møter motstand. Da vil kontrolleren gi beskjed til motorstyringen om at den møter motstand, og be den stoppe pådraget.



Figur 3: En modellering og inndeling av klasser for protesestyringsproblemet opp mot et scenario av proporsjonalstyring. Modellen til venstre er en videreutvikling av Losiers modell, som består av de tre hovedkategoriene liggende til venstre i figuren. Figuren til høyre er et scenario av simultan proporsjonalstyring. Bildekilde: Fougner et al. (submitted).

4 Målet med masteroppgaven

Målet med masteroppgaven er å se om BioPatRec-systemet er egnet til å overta for det nåværende systemet brukt på ITKs proteselaboratorium. I tillegg, gitt at systemet er egnet, begynne eventuelle implementasjoner av manglende funksjonalitet i det nye systemet kontra det nåværende systemet. Fokuset for denne oppgaven vil ligge på intensjonsestimeringen og systemtreningen rundt protesestyringsproblemet.

Analysen av BioPatRec-systemet skjer gjennom å sammenligne systemet opp mot en funksjonsspesifikasjon. Denne funksjonsspesifikasjonen adapteres fra tidligere arbeid gjort av Linnerud (2012), men utvides og modifiseres til å omfatte mer og omgjort funksjonalitet. En kan se denne funksjonsspesifikasjonen senere i opppgaven, se tabell 2 på side 28. Inni denne spesifikasjonen er alle krav som et potensielt protesesystem skal og bør kunne oppfylle. For å kunne gå videre fra analysen, må altså funksjonsspesifikasjonen være tilstrekkelig tilfredsstilt.

Fra analysen vil neste steg være å starte eventuelle implementasjoner av manglende funksjonalitet, som nevnt innledningsvis i dette kapitlet. Målet er å implementere intensjonsestimeringen av ITK-programvaren over i BioPatRec; da hovedsaklig simultan proporsjonalstyring siden BioPatRec allerede inneholder av/på-klassifiseringsalgoritmer. Det er også ønskelig å implementere PGT, som en ny type systemtrening, inn i BioPatRec.

Etter denne implementasjonen vil fortsatt ikke systemet være kjørbart med pådragsorganene på ITK-laben, men i utgangspunktet gjenstår bare outputdelen av implementasjonen. Målet i oppgaven av Berrum (2013) er inngangssignaler, fordi det meste av preprosesseringen er allerede implementert i BioPatRec. Den som setter sammen systemet vil likevel være nødt til å se at alt stemmer overens med hverandre, men et av målene i denne masteroppgaven er å gjøre denne overensstemmelsen relativt enkel.

Del II

Metode

Angrepsmåte i metoden

Metode-delen angripes ved å strukturere den etter målene med masteren:

Seksjon 5 Utstyr og funksjonsspesifikasjon viser til utstyret som er tilgjengelig og utvikler en funksjonsspesifikasjon, se seksjon 5. Her stilles kravene som BioPatRec må tilfredsstille for å være egnet som et alternativ til ITK-programvaren.

Seksjon 6 Programvurdering vurderer BioPatRec opp mot funksjonsspesifikasjonen, samt funksjonaliteten og klasseinndelingen i programmet. Hensikten er å analytisk gå gjennom BioPatRec, og ta en vurdering av programmet som helhet.

Seksjon 7 Implementasjon begynner på en implementasjonen, gitt at programmet tilfredsstiller funksjonsspesifikasjonen. I implementasjonen er det også viktig å strukturere fremgangsmåten. Metoden som brukes i denne masteroppgaven, er å basere seg på klasseinndelingen av proporsjonalstyringsproblemet i figur 3. Fokuset ligger på intensjonsestimeringen i scenarioet for simultan proporsjonalstyring.

5 Utstyr og funksjonsspesifkasjon

Dette kapitlet tar for seg metodikken for proteseforskningen gjort i laboratoriet på NTNU. Teksten i dette kapitlet vil være delvis adaptert fra metodikken i arbeidet av Bertnum (2012).

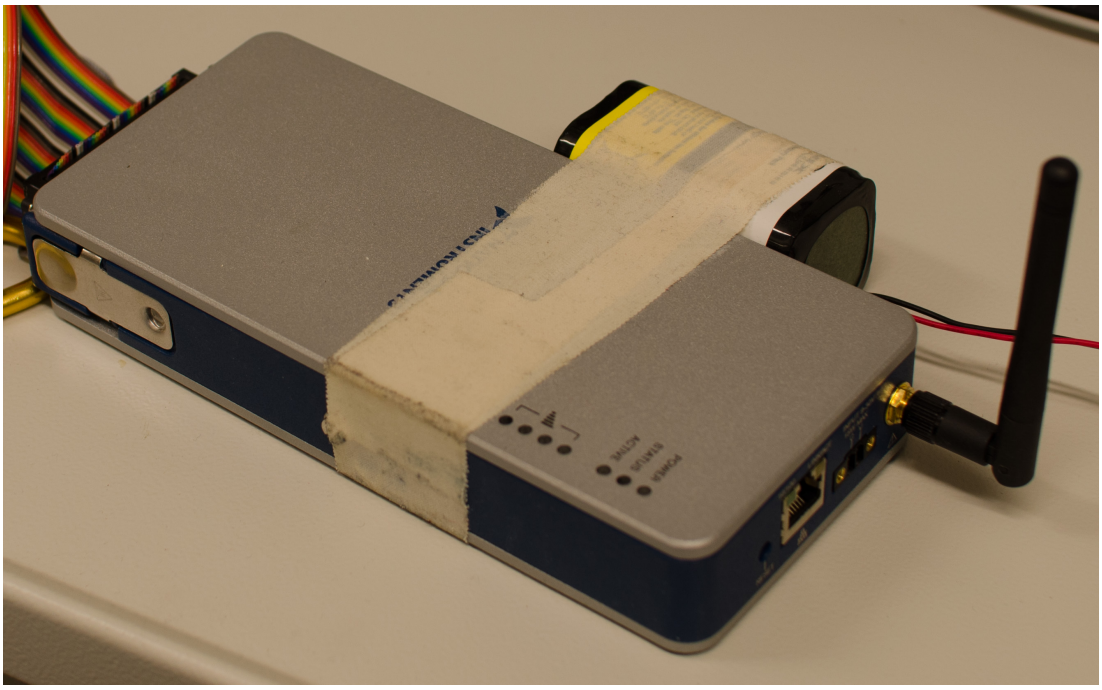
5.1 Utstyr

Til å måle EMG-signaler brukes trigno-sensorer fra Delsys Inc. (Boston, MA, USA), se figur 4. Disse kan måle SEMG på overflaten av huden, og har i tillegg innebygget akselerometer slik at posisjonsmåling er mulig gjennom beregning av orienteringen til sensorene i et tre-akset system.

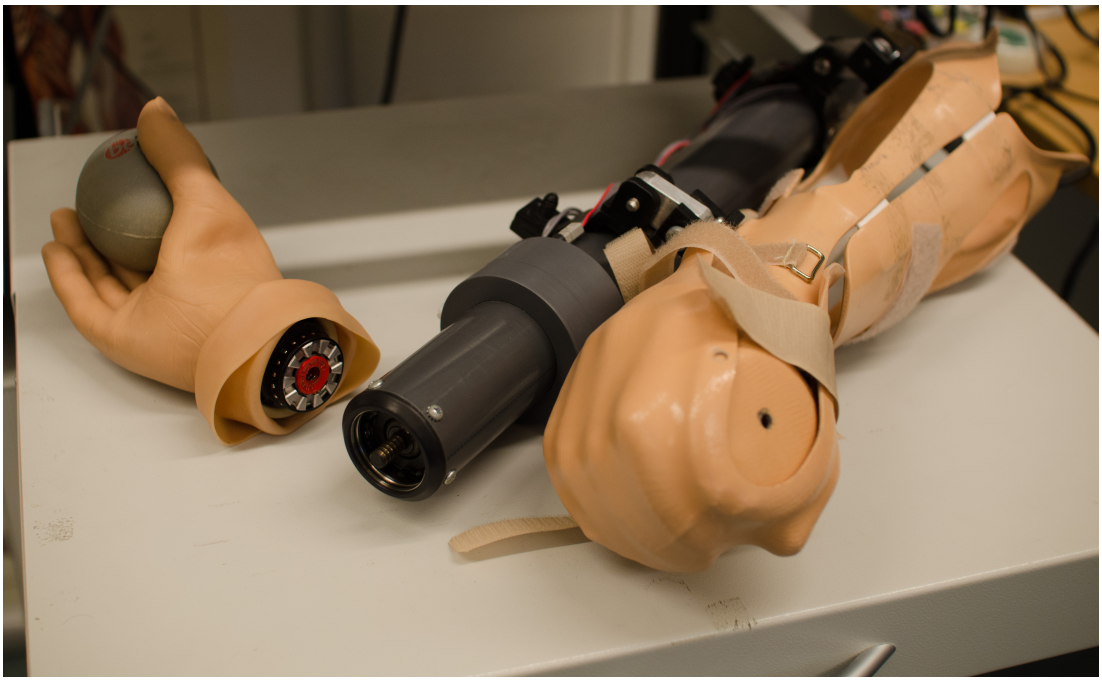
Figur 4: Et bilde av Trigno-sensorsystemet på ITK-laben. Modelldetaljer: *FCC ID*: W4P-SP-W01 (Sensor), W4P-SP-W02 (Basestasjon); *IC*: 8138A-DST01 (System)

Overføringen av EMG-signalene til systemet skjer via en trådløs overføring til basesentralen for Trigno-sensorene, deretter overføres de over USB til maskinen hvor protesestyringen foregår. Når dataene er behandlet, overføres de ved hjelp av en DAQ-enhet fra National Instruments (Austin, Texas, USA), se figur 5, til selve pådragsorganet, i dette tilfellet en MC-protese. Selve overføringen skjer over en trådløs kobling, og slik unngås rot i kabler samt restriksjoner som følger det å være kablet til en stasjon.

De to pådragsorganene som er tilgjengelig på ITK-laben, er nevnt i seksjon 3.3. Løsningen fra MC består av en silikonbelagt protesehånd med en børsteløs DC-motor, og et rotasjonsledd for håndleddet, se figur 6. Det andre pådragsorganet er robothånden som er illustrert i figur 7.



Figur 5: Et bilde av DAQ-enheten fra NI på ITK-laben. Modelnummer: cDAQ-9191



Figur 6: Bilde av en MC-protese og et rotasjonsledd, som er festet på en proteseholder. Protesemodellen heter ProHand og rotasjonsleddet heter ProWrist, og begge er levert av Motion Control.

Figur 7: Et bilde av robothånden på ITK-laben.

5.1.1 Programvare

Programmeringsverktøyet brukt for denne masteroppgaven er *Matlab* versjon 2012a og 2012b. Tidligere og fremtidige versjoner burde også fungere.

Det grafiske grensesnittet i *Matlab*-programmet er laget ved hjelp av *GUIDE* i *Matlab*, som er et innebygget grafisk verktøy. Den illustrerer hvordan grensesnittet ser ut, og vil tilby en rekke grafiske virkemidler en kan legge til i grensesnittet. Skulle en ønske å redigere grensesnittet er det bare å redigere filene generert av *GUIDE*. *Matlab* genererer *GUIDE*-filer i *.fig*-filer, som igjen genererer *Matlab*-kode i en *.m*-fil.

5.1.2 Proteseholder og sensorplassering

Slik som påpekt i arbeidet av Bertnum (2012), er det viktig å kunne simulere en faktisk protesebruker for å oppnå realistiske testresultater og treningsscenarioer. For å oppnå dette har Fougner et al. (submitted) designet og laget en proteseholder, se figur 8. På denne måten vil en bruker med begge hender kunne simulere en faktisk protesebruker. Ved å deaktivere funksjonen av den ene hånden er brukeren nødt til å bruke protesen for å kunne benytte seg av

denne hånden.

Figur 8: Illustrasjon av proteseholder og sensorplassering på en bruker. Figuren er lisensiert under en Creative Commons BY-NC-SA-lisens. Bildekilde: Fougner et al. (submitted)

Sensorplasseringen for sensorene sett i figur 8 er lik den i arbeidet av Fougner et al. (submitted), som videre også har blitt brukt i Bertnum (2012). Totalt er det fem EMG-elektroder som blir brukt, hvorav tre elektroder er plassert på den laterale siden av armen og to på den mediale siden. De tre på den laterale siden av armen er plassert på *m. abductor pollicis longus* (1), *mm. extensor digitorum & extensor digiti minimi* (2) og *mm. extensor carpi radialis longus & brevis* (3). Førstnevnte abduserer tommelen, neste strekker fingrene og sistnevnte supinerer underarmen. De to siste elektrodene, plassert på den mediale siden av armen, er posisjonert på *mm. flexor carpi radialis & flexor digitorum superficialis* (4) og *m. pronator teres* (5). Den første muskelen strekker på håndledd og fingre, og den siste sørger for pronasjon i underarmen.

Trigno-sensorene festes med en 4-hullet dobbeltsidig-tape levert fra Delsys. Denne sørger for at elektrodene har mest mulig kontakt med hudoverflaten. Dersom det er mye bevegelse og en dobbeltsidig-tape virker utilstrekkelig, kan det være lurt å gjøre forbindelsen sterkere med sports-tape eller lignende. På denne måten unngås plutselig avbrudd i koblingen som kan føre til hopp i målingene.

5.2 Funksjonsspesifikasjon

Før et nytt system skal lages eller at et nytt system skal adapteres, er det nødvendig at det oppfyller en funksjonsspesifikasjon. Slik sørges det for at all nødvendig funksjonalitet er tilstede i et potensielt nytt system. Det er også gunstig å legge til en liste over funksjoner som er ønsket implementert, men som ikke nødvendigvis er kritiske.

Til ITKs system vil tabell 2 være en representasjon på hvilke funksjoner som er nødvendige og ønskede i selve systemet. Denne spesifikasjonen er utviklet i samarbeid med Berrum (2013), og er basert på arbeid gjort tidligere av Linnerud (2012). Inndelingen er gjort slik at hver hovedklasse av krav representerer et av hovedstadiene i figur 3: *Preprocessing*, *Intent interpretation* og *Ouput*. I tillegg er det et eget klassekrav for *System training*, som er beskrevet i seksjon 3.2.4 og 3.4.

Krav R1 vil representere preprosesseringen, krav R2 omhandler intensjonsestimeringen, krav R3 tar for seg output og krav RS omfatter systemtreningen. Disse inndelingene står beskrevet i seksjon 3.4.

	Requirements	Description
Preprocessing	R1	The sensor input of the system **MUST** be modular and with the possibility to combine sensors.
	R1.1	Data from the Trigno base **MUST** be transferred through USB.
	R1.1.1	**MUST** function in real-time with at least 5xEMG and 5x3xACC. (5 Trigno sensors)
	R1.1.2	**SHOULD** be real-time connection for more then 5 sensors.
	R1.1.3	**SHOULD** be independent of third-party-software for reading data from USB.
	R1.2	Additional sensors **SHOULD** be easily implemented.
Intent interpretation	R2	The possibility of using different intent interpretations **MUST** be available.
	R2.1	All control systems **MUST** be modular to more easily implement new ones and to edit existing ones.
	R2.2	Pattern recognition **MUST** be a possible option as a control system. (Should probably be crisp classification since pattern recognition is quite a wide term.)
	R2.3	Proportional control **MUST** be a possible option as a control system.
	R2.3.1	**MUST** be possible to manipulate the prosthesis input to control the force, velocity and/or position.
	R3	The new prosthesis platform **MUST** be sufficiently compatible to function properly with the prosthesis setup that is currently at place in the prosthesis lab at NTNU.
	R3.1	The system **MUST** be able to use a communication protocol.

	Requirements	Description
Output	R3.1.1	It **SHOULD** be compatible with the NI DAQ devices.
	R3.1.2	It **SHOULD** be able to communicate through Bluetooth protocol.
	R3.1.3	It **MUST** work in real-time.
	R3.2	Continued use of the departments robot hand **SHOULD** be possible.
	R3.3	**MUST** work with the Motion Control prosthesis.
	R3.3.1	**MUST** work in real-time.
	R3.3.2	**SHOULD** be able to take advantage of all functions within the prosthesis.
System training	RS	**MUST** support alternative types of system training.
	RS.1	**MUST** support prosthesis guided training (PGT).
	RS.1.1	**MUST** have the possibility for exercise sets.
	RS.1.1.1	**MUST** support training through force-, velocity- or position-control.
	RS.1.1.2	**MUST** be optional to choose between the sets.
	RS.1.1.3	**MUST** be intuitive motions to follow.
	RS.2	**MUST** support screen guided training (SGT).

Tabell 2: Tabell med funksjonsspesifikasjonen for et potensielt protesesystem. Funksjonsspesifikasjonen er adaptert fra Linnerud (2012).

6 Programvarevurdering

For å begrunne starten av en implementasjon til det nye systemet, bør det ha vært en vurdering av den aktuelle programvaren. I forrige kapittel ble en funksjonsspesifikasjon designet akkurat for dette formålet, se tabell 2. Altså blir neste steg å foreta en vurdering av programvaren i henhold til funksjonsspesifikasjonen, samt funksjonaliteten og klasseinndelingen i programmet.

Kravene legges til i klammeform [R*] etter hvert som de avdekkes i vurderingen, hvor "*" representerer alle underinstanser. På denne måten blir det enkelt å relatere teksten til selve funksjonsspesifikasjonen.

6.1 Vurdering av preprosesseringen

I vurderingen av preprosesseringen, kan en se at støtten er der for en modularisert sensor-ordning [R1]. Likevel kreves det en del tilpasning for å implementere et sett med nye sensorer. I arbeidet av Berrum (2013) kan en se flere detaljer rundt vurderingen av sensorimplementasjonen, derav sensorene brukt i ITK-laboratoriet. Selv om det er ulemper i implementasjonen av sensorsystemet på ITK-laben, er fordelene større, derfor er kravene tilfredsstilt [R1.*].

For en sammenligning av egenskapene tilgjengelig i hver programvare, se figur 3. BioPatRec tilbyr en rekke egenskaper som kan trekkes ut av input-signalet, og derav flere som ikke eksisterer i ITK-programvaren. Uansett er det ønskelig å kunne implementere muligheten for å trekke ut de egenskapene som brukes i ITK-programvaren. På den måten vil det være mulig å fortsette med de samme eksperimentene i BioPatRec som de utført tidligere i ITK-programvaren.

	BioPatRec		**ITK-software**	
	Domain		Domain	
Features	*Time*	*Frequency*	*Time*	*Frequency*
Mean	•	•		
Mean absolute value	•		•	
Median	•	•		
Standard deviation	•			
Variance	•		•	
Waveform length	•	•	•	
RMS	•			
Zero-crossing	•		•	
Peaks (over RMS)	•			

	BioPatRec		**ITK-software**	
	Domain		Domain	
Features	*Time*	*Frequency*	*Time*	*Frequency*
Peaks mean	•			
Mean velocity	•			
Slope changes (peaks)	•			
Slope changes (diff)	•			
Power	•			
Difference abs. mean	•			
Max fractal length	•			
Fractal dimension	•			
Fractal dim. Higuchi	•			
Rough entropy	•			
Correlation	•			
Co-variance	•			
Peaks mean (Top 5)		•		
Peaks median (Top 5)		•		
Peaks std (Top 5)		•		
Autoregressive model			•	
Cepstral coefficients (EMGcc[1])				•
Cepstral coefficients (EMGcc2[2])				•
Histogram			•	
Myopulse percentage rate			•	
Number of turns			•	
Wilson amplitudes			•	
Average amplitude change			•	

Tabell 3: Sammenligningstabell av signalegenskapene tilgjengelig i BioPatRec og ITK-programvaren.

I preprosesseringen produseres et datasett kalt *xSets*, og den består av tre datasett: et treningssett, *trOuts*; et valideringssett, *vSets*; et testingssett, *tSets*. Disse er matriser på formen $r \times c$ (rader×kolonner), hvor r representerer antallet datasett, og c gir produktet av antall kanaler ganger antall egenskaper. De to første settene blir anvendt i offline-delen av *PatRec* med funksjonen *OfflinePatRecTraining* for å trene og eventuelt validere algoritmene. Testingssettet blir behandlet av *OneShotPatRec*-funksjonen for å teste hvor godt algoritmene presterer, se seksjon 3.1.4 for detaljer rundt test-algoritmene.

[1] En av to måter i ITK-programvaren for beregning av cepstral koeffisientene.
[2] Andre måten i ITK-programvaren for beregning av cepstral koeffisientene.

Hver av settene i *xSets* har igjen tilhørende sett i *xOuts*: *trOuts*, *vOuts* og *tOuts*. Datasettene er på matriseformen $r \times c$ (rader×kolonner); her utgjør r antall datasett og c antall outputer. Foreløpig er det bare støtte for verdiene 0 og 1 i *trOuts*-, *vOuts*- eller *tOuts*-matrisene, 0 for en *av*-output og 1 for en *på*-output. Disse verdiene forteller hvilken bevegelse som tilhører dataene i trening-, validering- eller testing-settene, altså *trSets*, *vSets* eller *tSets*. Slik vet en hvilken type bevegelse de ulike dataene tilhører.

Det ønskelige vil være å gjøre disse til verdier fra 0 til og med 1, for slik å oppnå skalerbare verdier. Sånn kan en få proporsjonalstyring på den måten det er konstruert i ITK-programvaren. Dette må eventuelt implementeres i systemtreningen, hvor det inkluderes opptak av data med støtte for proporsjonaltrening. Det mest naturlige er å gjøre dette gjennom PGT.

Problemet med *xOuts* er måten det beregnes på, for dette er ikke data som kommer direkte fra opptaket av data. Beregningen skjer ut ifra rekkefølgen på bevegelsene i datasettene. Rekkefølgen står i henhold til rekkefølgen på opptakene, altså kan en segmentere dataene basert på denne rekkefølgen. Ut ifra dette tilegnes en vektor med 0 eller 1 i *xOuts* til hvert av settene *xSets*.

Det er to mer innlysende løsninger på problemet: En er å gjøre om funksjonen som lager *xOuts* til å kunne ta hensyn til opptaksdata hvor proporsjonaldata er tilstede. Den andre metoden er å legge til en ekstra variabel i *struct*-tabellen, og deretter benytte seg av den i algoritmene som støtter det. Den andre løsningen er nok den som krever minst arbeid, og i tillegg ikke ødelegger for gamle metoder. I fremtidig arbeid er det foreslått en måte for å implementere denne funksjonaliteten.

6.2 Vurdering av intensjonsestimeringen

Det er full mulighet for å benytte forskjellige implementasjoner av intensjonsestimering i BioPatRec [R2]. Alle disse befinner seg i den delen av programmet som heter *Pattern recognition*, og tilsvarer det som i figur 3 kalles *Intent interpretation*, eller på norsk, intensjonsestimering.

Nye systemer er enkle å implementere så lenge de behandles likt som i BioPatRec [R2.1, R2.2], og det står detaljert på BioPatRec's hjemmesider hvordan nye algoritmer skal implementeres (Ortiz-Catalan, n.d.). Normalt sett vil dette fungere ved at en oppretter to funksjoner: en for treningen av algoritmen og en for å teste den trenede algoritmen.

Algoritmetrenings-funksjonen mottar det som i *Matlab* kalles en *struct*, som kan forstås som en strukturert tabell bestående av utvalgt informasjon. Denne strukturerte tabellen inneholder data som er behandlet så langt i programmet,

altså det behandlede input-signalet. Alt av hva den inneholder er ikke relevant å bringe frem, men det kan undersøkes nærmere på wiki-sidene hvis ønskelig (Ortiz-Catalan, n.d., Søkeord: sigFeatures structure). Det som er viktig å påpeke er at den foreløpig ikke har en støtte for proporsjonaldata, så dette må eventuelt implementeres på et punkt i datastrukturen.

I treningen av algoritmen skjer de forhåndsberegningene som må gjøres før en algoritme kan kjøres i sanntid. Denne prosedyren varierer fra algoritme til algoritme, men resultatet er et operativt system. Ved kjøring i sanntid, vil den valgte algoritmen kunne benyttes opp mot alle tilgjengelige pådragsorganer.

For at protesestyringen skal være kjapp nok til å være intuitiv å styre, må beregningene skje i sanntid. Dette krever at algoritmene som benyttes tilfredsstiller et tidskrav. Dermed begrenses utvalget av algoritmer som kan benyttes relativt med maskinvaren som gjør beregningene. Jo kraftigere maskinvare, desto flere algoritmer kan benyttes, og vice versa. Kravene vil også skalere med maskinvaren, lavere krav for god maskinvare og høyere for dårlig maskinvare.

I utgangspunktet er det støtte for at algoritmer kan testes i BioPatRec, derav den andre funksjonen som tester den trenede algoritmen. Ved en slik testing beregner programvaren hvilken bevegelsesklasse som blir utfallet av en spesifikk bevegelse. Ytterligere vil den gi sannsynlighetene for prediksjonen av de andre bevegelsesklassene. Dette forteller noe om kvaliteten på input-dataen til algoritmen, og kan være en pekepinn på oppførselen til et eventuelt pådragsorgan.

Foreløpig eksisterer ikke proporsjonalstyring som et styresystem i BioPatRec. ITK-programvaren har i tillegg en egen måte utføre denne styringen på, og den stemmer ikke overens med det som skjer i BioPatRec. Derfor kreves det mer arbeid for å implementere proporsjonalstyring kontra en standard mønstergjenkjennings-algoritme.

Problemet med en eventuell implementasjon av proporsjonalstyring i BioPatRec, er at inputen til proporsjonalstyring er i form av motorfunksjoner, ikke bevegelsesklasser som brukes ellers i BioPatRec. Derfor bør det opprettes en metode for å gjøre oversettingen mellom disse input-formatene. Ideelt sett vil en kunne gjøre denne oversettingen begge veier, for da kan en konvertere formatene både før og etter en algoritme.

Fordi intensjonsestimeringen for proporsjonalstyring kjøres på en egen måte i ITK-varianten, innebærer det å innføre et eget grensesnitt for selve bruken av styringen. Slik blir det mer enkelt å holde modulariteten i programmet siden en ikke trenger å gjøre store modifikasjoner på eksisterende grensesnitt [R2.3].

6.3 Vurdering av outputen

Hvis det utvikles en kommunikasjonsmetode for MC-protesen på NTNUs protese-lab, vil denne protesen fungere i BioPatRec [R3, R3.3]. Dette kan utføres ved hjelp av en DAQ fra NI, slik som det er gjort tidligere i ITK-programvaren, og BioPatRec-programvaren [R3.1, R3.1.1]. Likevel er det benyttet forskjellig DAQ-enheter, så det blir noen forskjeller i kommunikasjonen med kommunikasjonsprotokollen. Et alternativ er å utvide eksisterende kommunikasjonsprotokoll med blåtann-kommunikasjon fordi MC-protesen har støtte for dette [R3.1.2]. Dette ville skapt en trådløs og mer mobil løsning for protesestyring, og eventuell bruker- og systemtrening. Uansett er dette valg som må tas ved en eventuell implementasjon, og begge er mulige alternativer for kommunikasjon med protesen.

Forrige avsnitt gjelder også for instituttets robot-hånd [R3.2]. Siden behandlingen av input-signal allerede er tilgjengelig med forskjellige alternativer, må det bare utvikles en kommunikasjonsmetode for å styre hånden.

Både for MC-protesen og robot-hånden hos ITK, vil det være fullt mulig å utnytte protese-funksjonaliteten i begge to [R3.3.2]. Alle bevegelsene er allerede tilgjengelig i BioPatRec, altså må det bare inkluderes i implementasjonen av kommunikasjonsmetodene til begge pådragsorganene.

Slik som nevnt i kapittel 6.2, i vurderingen av intensjonsestimeringen, er det maskinvaren som begrenser for sanntidsberegninger. Flaskehalsen vil sjelden oppstå i kommunikasjonen mellom systemet og pådragsorganet, gitt at kommunikasjonen støtter den nødvendige hastigheten som kreves. Derfor vil de fleste beregninger kunne skje i sanntid hvis hastighetskravet er oppfylt [R3.1.3, R3.3.1].

6.4 Vurdering av systemtreningen

Det er allerede alternativer for systemtrening tilgjengelig i BioPatRec, men alle går under kategorien SGT [RS, RS.2]. Dette er ikke uventet med tanke på at PGT er en såpass ny metode for systemtrening. Felles for begge er at systemet trenes opp ved at input-signalet avbildes opp mot output-signalet. Dette står mer om i seksjon 3.2.4 om systemtrening.

Systemtreningen i BioPatRec er så langt konstruert slik at alle målinger registreres som av eller på. Det måles altså én styrke fra brukeren. Dette stemmer imidlertid ikke overens med slik det fungerer i ITK-systemet. ITK-systemet har basert seg på å måle input-signalet proporsjonalt med styrken fra brukeren. For å gjøre dette er en avhengig av å vite hvordan treningssettet

fungerer, altså trenger en mer kunnskap fra opptaket av data enn det i den eksisterende løsningen. Hvis en skal bruke proporsjonalstyring slik som i ITK-programvaren, må en altså inkludere mer informasjon fra et treningssett ved en implementasjon.

Hvis PGT implementeres i BioPatRec, må det gjøres fra bunnen av, men støtten for det vil være der [RS.1]. Ved en slik implementasjon vil en da kunne tilrettelegge slik at alle krav i tabell 2 er tilfredsstilt [RS1.1, RS1.1.1, RS1.1.2, RS1.1.3].

I en implementasjon av PGT vil en kommunisere med pådragsorganet, i dette tilfellet MC-protesen, for at det i hele tatt skal være noe poeng i å implementere PGT. Derfor bør dette skje i samarbeid med output-implementasjonen, og det er dessverre ikke fokuset i denne masteroppgaven. Altså er dette et tema i fremtidig arbeid.

6.5 Konklusjon av programvarevuderingen

Etter en vurderingen av de ulike bolkene i BioPatRec, og i konsensus med Kristian Berrum (Berrum, 2013), vil en implementasjon være forsvarlig å påbegynne. Selv om mye funksjonalitet krever en del jobb å implementere, er det fortsatt et mer fremtidsrettet program med tanke på modularitet. I tillegg er selve prosjektet et åpent kildeprosjekt, så nye bidrag vil sannsynligvis forekomme hyppigere. Videre forskning blir også lettere siden de som benytter seg av det kan ha et mer fokusert forskningsområde. Nye deltakere av prosjektet er ikke avhengig av å starte fra bunnen av, og de slipper derfor å jobbe seg oppover i oppbyggingen av et slikt protesestyringssystem. I tillegg vil deling av resultater og erfaringer bli lettere når alt er basert på en felles forskningsplattform.

Når det kommer til detaljer rundt funksjonsspesifikasjonen, er det ingen krav som ikke er mulig. Noen krav vil kreve litt jobb, men kan potensielt sett ha et fremtidsrettet preg på programmet. Begrunnelsen for dette utsagnet er at ny funksjonalitet gir økte muligheter for videre eksperimentering. For eksempel vil en implementasjon av PGT som systemtreningen åpne for flere typer eksperimenter. Dette kan igjen føre til en videreutvikling av systemtrening, da spesielt PGT, eller algoritmene for intensjonsestimering som testes med ulike typer systemtrening.

7 Implementasjon

Basert på kapittel 6 i vurderingen av BioPatRec, kan en implementasjon forsvares. Derfor vil dette kapitlet ta for seg detaljer og beskrivelser rundt implementasjonen av ITK-funksjonaliteten over i BioPatRec.

I tabell 5, under seksjon 9 i del III, vises alle de funksjonene som er implementert i BioPatRec under denne masteroppgaven, i tillegg litt informasjon vedrørende funksjonene og hvilke funksjoner som kaller dem. Underseksjonene i denne seksjonen vil ta for seg detaljer rundt funksjonene som er implementert og opplistet i denne tabellen.

For å gi en oversikt over endringer gjort i eksisterende funksjoner, er det laget en tabell som inneholder hvilke endringer som er gjort og hvor i funksjonene endringer er gjort, se tabell 6 under resultater i del III. Slik vil det være lett å se hva som er nytt og forandret, eventuelt hvorfor disse forandringene er gjort.

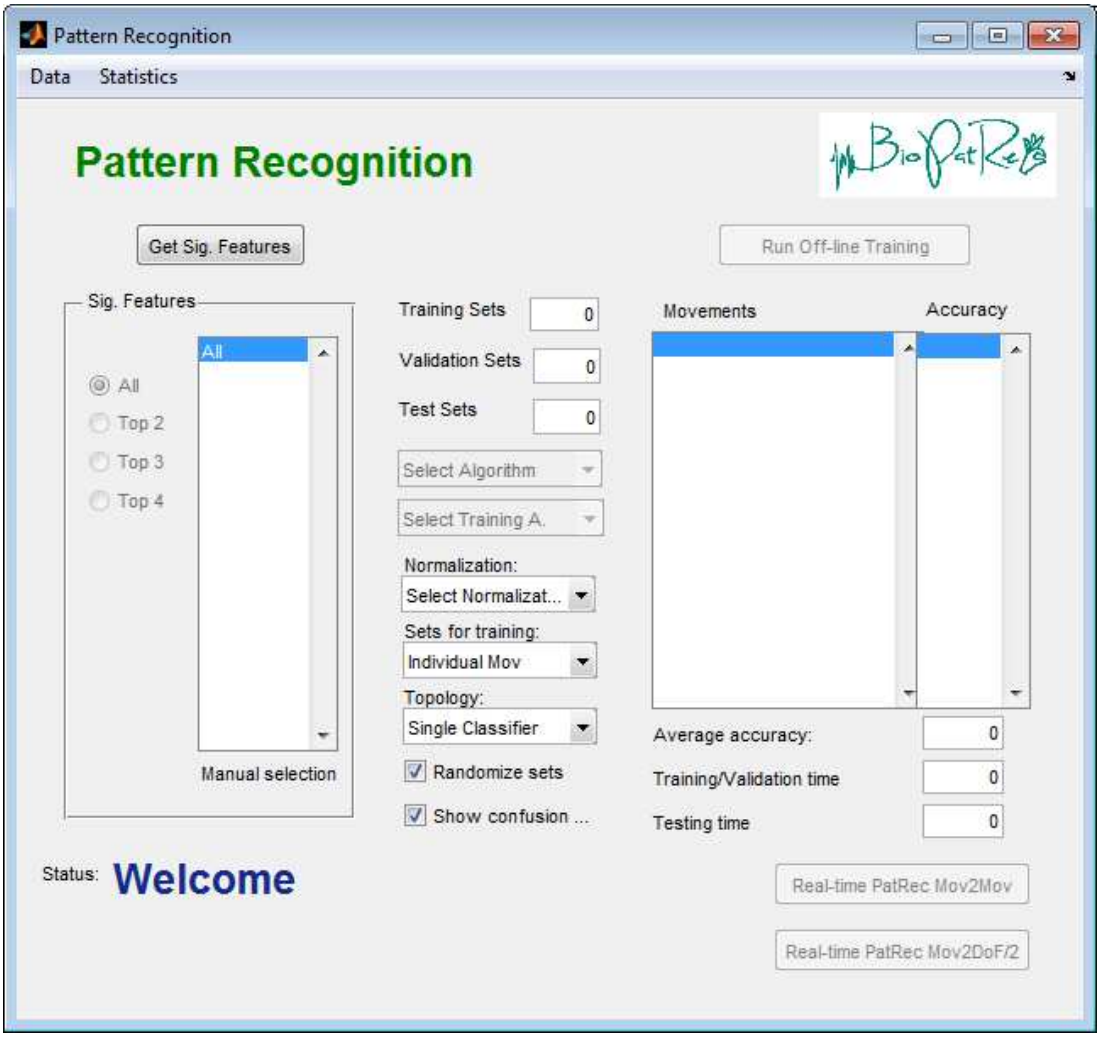
7.1 Offline-proporsjonalstyring

I BioPatRec-figuren (fig. 9) er det en knapp som heter *Run Off-line Training*. Her skjer de nødvendige kalkulasjonene som må til før selve proporsjonalstyringen kan kjøres i sanntid. Denne vil ta utgangspunkt i hvilken algoritme som er valgt, samt hvilke behandlingsmetoder som er valgt.

For proporsjonalstyring er det to estimatorer tilgjengelig for å trenes opp og kjøres i sanntid; *Normal linear estimator* og *Decoupled linear estimator*. Disse algoritmene er dokumentert i arbeidet av Linnerud (2012). Første steg for estimatorene er å regne ut matrisen som avbilder input-signalene opp mot output-signalene. Dette skjer basert på målingsdata tatt opp i forkant, derav offline-trening. Etter det skal algoritmene kjøres i sanntid. Offline-estimatorene er implementert i funksjonen kalt *OfflineProportionalControl*, se tabell 5 under seksjon 9 i del III. Sanntids-estimatorene er implementert i *LinearMapping*-funksjonen, se tabell 5.

På grunn av måten proporsjonalstyring behandler dataen i *xSets*, må en stille noen krav til dem når de skal gjennom offline-treningen for proporsjonalstyring. Kravene forteller hvilke bevegelser som må være tilstede for at treningen skal kunne utføres. Dersom de ikke er til stede, vil det gis en feilbeskjed, og treningen vil ikke utføres. Dette er en midlertidig løsning for å implementere den funksjonaliteten som er i ITK-programvaren.

Problemet i implementasjonen, er at BioPatRec-algoritmene har et annet format på input-/output-data sammenlignet med ITK-softwaren. Løsningen på



Figur 9: Illustrasjon av BioPatRec's mønstergjenkjenning GUI. Bildekilde: Fra programvaren i Ortiz-Catalan et al. (2013)

dette var å omformatere input-data, altså *xSets*, som kom inn i algoritmen slik at det passet med de gamle algoritmene.

Fremgangsmåten som brukes for å omformatere *xSets*, er å søke etter de spesifikke bevegelsesklassene, som til sammen definerer en frihetsgrad, eller en motorfunksjon. I dette tilfellet betyr det å lete etter åpning og lukking av hånden, som definerer en frihetsgrad, og deretter finne pronasjon og supinasjon av underarmen, som definerer enda en frihetsgrad. Dette er mulig på grunn av *mov*-variabelen, som inneholder hvilke bevegelsesklasser som omfatter dataene i *xSets*. *mov*-variabelen er en av inputene til *OfflinePatRecTraining*-funksjonen, som initialiserer *OfflineProportionalControl*. Ordene bevegelsesklasse og motorfunksjon, står listet opp i tabell 4 med en beskrivelse og eksempler.

Ved å dele bevegelsesklassene inn i frihetsgrader, vil de fungere som motorfunksjoner for et pådragsorgan, hvor – i dette tilfellet – positive verdier definerer én eller en kombinasjon av bevegelse(-r), og negative verdier definerer antagonist-bevegelsen(-e). Metodisk er det en naturlig måte å kommunisere med et pådragsorgan, men det krever, som påpekt tidligere, en annen kommunikasjonsmetode for å behandle data i sluttfasen.

For øyeblikket er det bare datasettet *xOuts*, som forteller med verdiene 0 og 1 hvilket datasett som tilhører hver av bevegelsesklassene. Ønsket er å ha verdier som skalerer fra 0 til og med 1, slik at en får proporsjonaldata inn i algoritmetreningen. For å unngå og ødelegge eksisterende metoder, bør en ikke endre *xOuts* for å legge til denne dataen. Forslaget er å lagre proporsjonaldata i en ny variabel i *struct*-tabellen fra *recSession*-delen i BioPatRec.

Fordi en endring i input-format gir en endring i output-format, er en nødt til å endre sluttbehandlingen av data siden den ikke stemmer overens med den i BioPatRec. Dette ble løst ved at en sender output-dataene direkte til et eget grensesnitt, *GUI_ProportionalControl*, og dermed benytter seg av en annen databehandling i sluttfasen. Når brukeren trykker på knappene *Real-time PatRec Mov2Mov* eller *Real-time PatRec Mov2DoF/2* i figur 9, så sendes altså brukeren direkte til *GUI_ProportionalControl*. Proporsjonalstyring benytter seg altså ikke i det hele tatt av de originale grensnittene for databehandling i sluttfasen. Grensesnittet for proporsjonalstyring kan ses i figur 10 i seksjon 7.2.

Ligning 7.1 for lineær regresjon er en omskriving av ligning 3.2 i seksjon 3.2.2. I *Matlab* er ligning $(\vec{x}'\backslash\ \vec{F}')' = \vec{F}/\vec{x}$, gitt at formen på vektorene stemmer overens, men kjøretiden virker å være noe mindre for ligning 7.1. Derfor benyttes formlene 7.1, 7.2, 7.3 og 7.4 i implementasjonen til BioPatRec. Noe av grunnen for at "\"-operatoren ble brukt tidligere, er at den inneholder en innebygget LSQR-metode. Dette gjelder også for "/"-operatoren, bare at ligning 7.1 reduserer kjøretiden i forhold til ligning 3.2.

Den nye ligningen for estimeringen av $\mathbf{A}$ blir for normal lineær estimatoren som i ligning 7.1.

$$\mathbf{A} = \vec{F}/\vec{x} \tag{7.1}$$

For den dekoblede lineære estimatoren, blir formlene slik om i ligning 7.2, 7.3 og 7.4.

$$\mathbf{A_{mean}} = \vec{F_{mean}}/\vec{x} \tag{7.2}$$

$$\mathbf{A_{std}} = \vec{F_{std}}/\vec{x} \tag{7.3}$$

$$\mathbf{A_{form}} = \vec{F_{form}}/\vec{x} \tag{7.4}$$

I praksis brukes bare den normal lineære estimatoren. Dette skyldes at den gir tilnærmet like god oppførsel som den dekoblede lineære estimatoren, men er mye raskere å beregne. Siden den dekoblede lineære estimatoren var kjapp og enkel å implementere, er den der likevel som et alternativ og som inspirasjon for å videreutvikle tanken. I seksjon 3.2.2 nevnes dekoblet *ulineær* estimator som en potensiell tanke, og dette er et mulig tema for fremtidig arbeid.

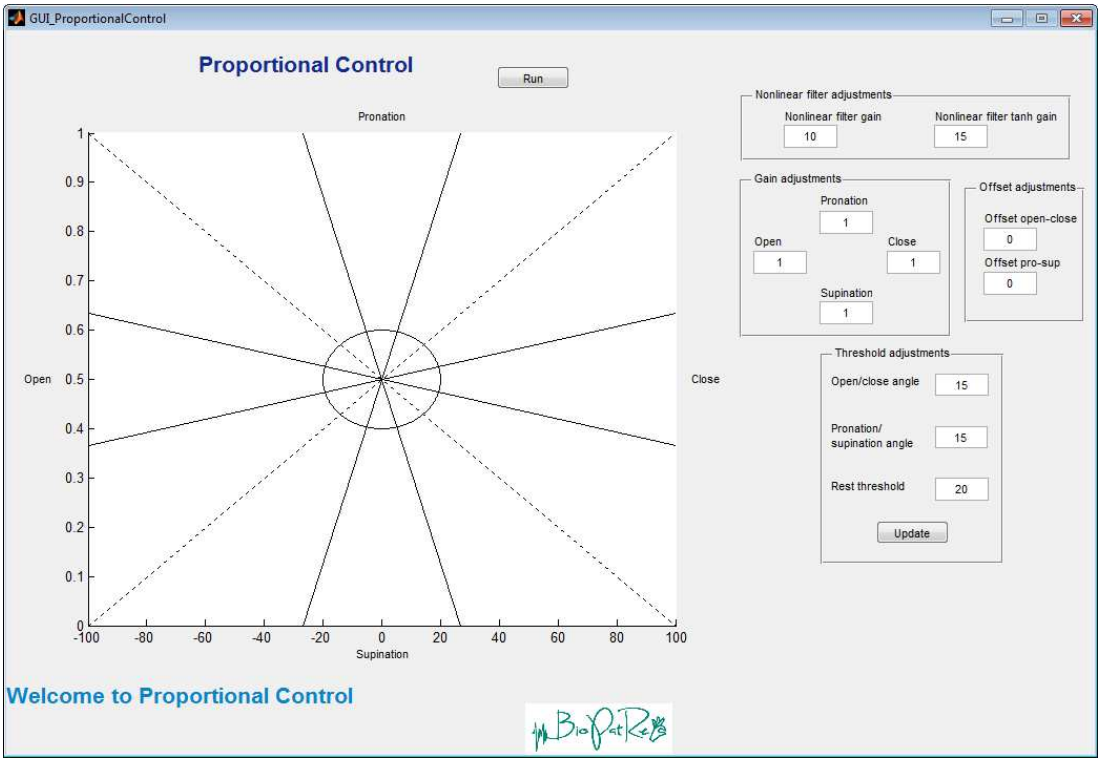
I offline-delen ble det også lagt til en funksjon kalt *ProportionalControlTest*, se tabell 5. Hensikten med denne skal være å teste hvordan algoritmen presterer, altså hvor godt proporsjonalstyringen faktisk fungerer. Problemet er at algoritmene i algoritmetreningen behandler dataen på en annen måte. Derfor må en mest sannsynlig designe denne funksjonen på en annen måte en det som er gjort tidligere. For øyeblikket er denne funksjonen bare en tom funksjon som setter utgangsverdiene lik null. Slik kan i alle fall programmet kjøres, men en vil ikke få et estimat på hvor godt algoritmene presterer. I fremtidig arbeid, er det lagt frem et forslag til hvordan denne funksjonen kan lages, se seksjon 12.

7.2 Grensesnittet for proporsjonalstyring

Dette bidraget til prosjektet baserer seg på et kjørbart grensesnitt for å håndtere en protese med proporsjonalstyring. Brukeren benytter seg av grensesnittet ved å justere de ønskede parametrene, for så å kjøre det i sanntid når alt er klart. Det er også mulighet for å justere parametrene mens kjøringen foregår. Grensesnittet er i stor grad basert på det gjort i forrige programvare av Linnerud (2012), og kan ses i figur 10. Funksjonaliteten i grensesnittet ligger i funksjonen *GUI_ProportionalControl*, hvorav en *.m*-fil inneholder koden som blir kjørt, mens en *.fig*-fil inneholder det grafiske grensesnittet. Denne funksjonen er oppført i tabell 5.

I figur 10 ser en det grafiske grensesnittet. Her er det mulig å justere tersklene illustrert i figuren. Dette kan gjøres ved å endre verdiene i *Threshold adjustments*-vinduet, og deretter trykke på *Update*-knappen. Resultatet av dette vil være at aksene i figuren fysisk endres. Endringene vil også påvirke proporsjonalstyringen, da spesielt justeringen av forsterkning og terskelverdier som bruker terskelverdiene som parametre i utregningene. Denne justeringen vil bli beskrevet senere i seksjon 7.2.4.

Under-seksjonene i denne seksjonen tar for seg funksjonene som er implementert i dette grensesnittet. Funksjonene er kort oppsummert i tabell 5, men redegjøres mer for i kommende under-seksjoner. For et overblikk over funksjonshierarkiet henvendes leseren til figur 11.

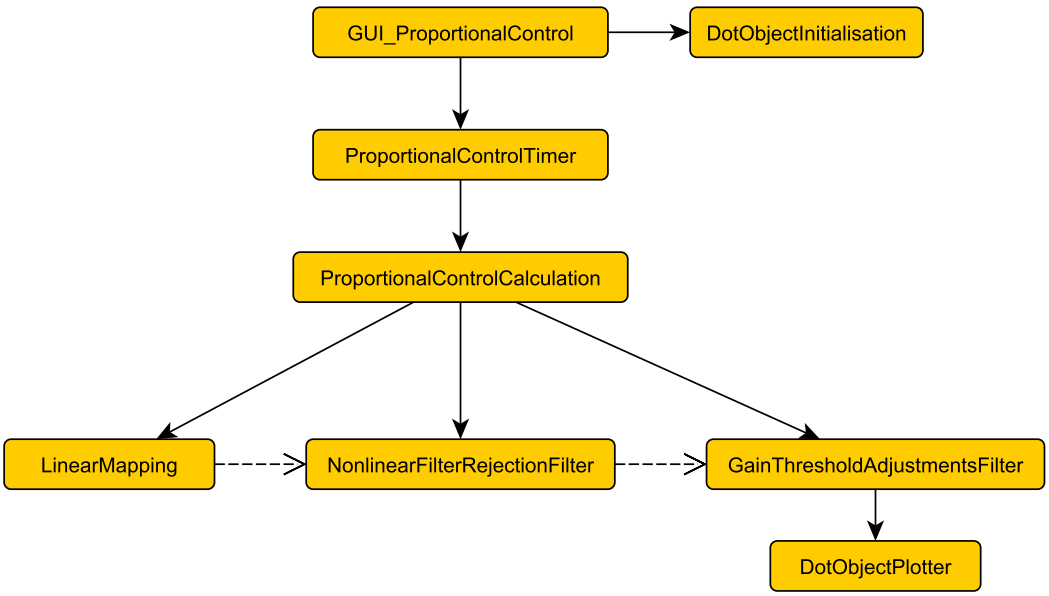


Figur 10: Illustrasjon av GUIen til proporsjonalstyringen implementert i BioPatRec.

7.2.1 Sanntids-proporsjonalstyring

Ved å trykke *Run*-knappen i figur 10, trigges *ProportionalControlTimer*-funksjonen som igjen initialiserer et *timer*-objekt med funksjonen *ProportionalControlCalculation*. Dette starter sanntidskjøringen av proporsjonalstyring. *Timer*-objektet som startes i *Matlab* vil kjøres som en egen tråd i bakgrunnen. Etter hvert som input-data kommer inn til proporsjonalstyringen, vil det plottes i sanntid langs aksene i figur 10. Funksjonene nevnt i dette avsnittet står opplistet i tabell 5.

Siden *timer*-objektet alltid sjekker parametrene for hver kjøring i tråden, vil den justere reguleringen etter som de justeres. De ulike parametrene som kan justeres illustreres i figur 10. Dette vil enten være å justere terskel-, forsterkning- eller offsetverdiene, eller å justere de ulineære filterforsterkningene.



Figur 11: Illustrasjon av funksjonshierarkiet til proporsjonalstyrings GUIen.

Ved å justere tersklene blir operasjonsområdene for å styre en bevegelse større eller mindre, men på bekostning av eller til fordel for simultan styring. Jo høyere terskelverdiene er for å gjøre én type bevegelse, altså et større areal, desto mindre er området for å gjøre en simultan styring. Dette står mer detaljert beskrevet i paragrafen om justering av forsterkning og terskelverdier i seksjon 3.2.2. Disse hendelsene kommer av at endringer i terskelverdiene direkte påvirker denne justeringen, se seksjon 7.2.4, som også påpekt innledningsvis i seksjon 7.2.

Forsterkningen vil være et verktøy for å hjelpe brukeren til å nå de høyeste pådragene, som påpekt i seksjon 3.2.2 under paragrafen om justering av forsterkning og terskelverdier. Dette er nyttig og praktisk å kunne gjøre i sanntid, for det er ønskelig av brukeren å gi fullt pådrag uten å kontrahere muskler til grensen av ubehag.

Dersom brukeren ser et mønster i at hviletilstanden hviler langt unna senteret av plottet, så kan offset-parametrene justeres for å sette dette punktet nærmere origo. En slik funksjonalitet vil kompensere for dårlige målinger. Hvis brukeren i tillegg bruker protesen over lang tid, slites musklene ut og en kan få en endring i bevegelsesmønsteret. Da kan en justering av offset-verdier sentrere oppførselen i bevegelsesmønsteret til origo av plottet.

Innledningsvis i denne seksjonen påpekes bruken av *timer*-objekt i *Matlab* for å kjøre proporsjonalstyring. Ved å bruke *Matlab*-funksjonen *timerfind*, kan

en lokalisere alle kjørende *timer*-objekter. Da vil en få en oversikt over alle de trådene som er opprettet og som kjører eller ikke kjører. Hvis et objekt videre merkes med et navn ('Name'), som fungerer som en merkelapp, kan en enkelt gjøre et spesifikt kall til dette *timer*-objektet. Slik er det lett å kontrollert stoppe og eventuelt slette det spesifiserte objektet. Implementasjonen av dette står i funksjon *ProportionalControlTimer*, listet opp i tabell 5. I listing 1, kan en se funksjonen *ProportionalControlTimer* som initialiserer *timer*-objektet.

Listing 1: Initialisering og start av *timer*-objektet i *ProportionalControlTimer*-funksjonen.

```
1 function ProportionalControlTimer ( featureVector, handles )
2
3 %% Initialising and starting the timer function
4 T = timer ( 'TimerFcn', ...
      @(~,~)ProportionalControlCalculation(featureVector, handles), ...
      'Period', 0.5, 'ExecutionMode', 'fixedRate', 'Name', ...
      'ProportionalControlCalculation' );
5 start(T);
6
7 end
```

En svakhet med *timer*-objekt i *Matlab*, er at *Matlab* ikke anbefaler[3] det til sanntids-bruk. Dette på grunn av at begrensninger i maskinvare, operativ system eller programvare, kan påvirke effektiviteten. Til tross for dette er maskinvaren på ITK-laben, per dags dato, tilfredsstillende nok til å benytte denne funksjonaliteten. Dette er likevel lurt å ha i bakhodet hvis det senere blir problemer med treghet i programmet. I så fall må en se etter en alternativ metode.

Funksjonen som innkapsler selve intensjonsestimeringen, kalles *ProportionalControlCalculation*. Her skjer selve sanntids-kalkulasjonen for proporsjonalstyringen. Hvilken funksjonalitet som foregår her, er beskrevet i bakgrunnen under seksjon 3.2.2, og seksjonene videre vil ta for seg implementasjonen av denne funksjonaliteten.

7.2.2 Lineær mapping

Funksjonen *LinearMapping*, se tabell 5, tar inn to inputs, og returnerer en vektor med et estimat på pådraget til pådragsorganet. En av input-variablene er en *handle*, som er et håndtak til en mengde med satte variabler, og den andre er en egenskapsvektor som kontinuerlig oppdateres i sanntid. Det

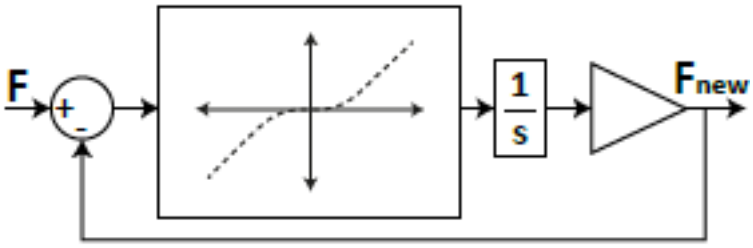
[3] Se notatet under "doc timer" i *Matlab*

eneste håndtaket brukes til er å hente ut A-matrisen, på forhånd utregnet i *OfflineProportionalControl*-funksjonen, og hvilken type estimator som er brukt.

For å skille mellom hvilken type estimator som brukes, sjekkes variabelen med estimator-typen. Deretter behandles egenskapsvektoren slik som den utvalgte algoritmen tilsier, og produserer så et output-estimat. Beregningen av dette estimatet står nærmere beskrevet i seksjon 3.2.2 under paragrafen om normal linear estimering, og i seksjon 3.2.2 under paragrafen om dekoblet lineær estimering.

7.2.3 Ulineær støydemping

Designet bak dette filteret er beskrevet i paragrafen om ulineær støydemping under seksjon 3.2.2. Modellen kan ses i figur 12. Inputen til funksjonen blir altså estimatet fra den lineære avbildningen, samt den forrige verdien fra den ulineære støydempingen. Ut av funksjonen kommer et nytt estimat på pådraget fra funksjonen *LinearMapping*, se tabell 5.



Figur 12: Illustrasjon av den ulineære støydempingen. Bildekilde: Utsnitt fra bilde i (Fougner, 2013, s. 112)

Hensikten med *NonlinearFlutterRejectionFilter*-funksjonen er å motvirke hurtige endringer i pådraget til pådragsorganet. Hvis ikke denne filtreringen kjøres, risikeres det at pådragsorganet starter å vibrere på grunn av hyppige endringer i pådraget. Dette kommer av at de bioelektriske-signalene, ved EMG-måling, oscillerer. Selv om brukeren teoretisk sett gjør den samme bevegelsen, tilsier signalene at det skjer mye bevegelser. Resultatet av filteret er at pådraget holdes konstant ved å filtrere bort oscillasjoner i målingene.

Til å huske estimatet fra forrige funksjonskall, brukes det som i *Matlab* kalles en *persistent*-variabel. Hensikten er å ha en lokal variabel som lagres i minne og kan kalles i senere funksjonskall; i tillegg er variabelen bare tilgjengelig i funksjonen den initialiseres. Derfor vil en ikke få et problem med at andre deler av programmet kaller på en variabel med samme navn.

For å spare maskinkraft er integralet i filteret forhåndsberegnet og hardkodet inn i funksjonen. Dermed trenger en bare å sette inn input-variablene for å få outputen.

7.2.4 Justering av forsterkning og terskelverdier

Denne justeringen tar inn det så langt estimerte pådraget, sist behandlet av *NonlinearFlutterRejectionFilter*-funksjonen, og benytter forsterkning, offset- og terskelverdiene, satt i GUIen til å gi et mer nøyaktig pådrag til pådragsorganet, se figur 10 for GUIen. Dette er beskrevet nærmere i paragrafen om justering av forsterkning og terskelverdier i seksjon 3.2.2.

GainThresholdAdjustments-funksjonen, se tabell 5, er i utgangspunkt designet slik at den har støtte for flere enn to frihetsgrader. Likevel, siden systemet i rundt den ikke har denne støtten ennå, benyttes bare justeringen av forsterkning og terskelverdier for tilfeller hvor antallet frihetsgrader er lik to. Koden bak denne justeringen er adaptert fra tidligere arbeid (Linnerud, 2012).

7.2.5 Generering og oppdatering av plotte-objektet

I plottet er det et plotte-objekt som viser målingene en får i sanntid. Dette objektet genereres av funksjonen *DotObjectInitialisation*, listet i tabell 5, og blir kalt før *timer*-objektet initialiseres. Selve oppdateringen av plotte-objektet skjer under justeringen av forsterkning og terskelverdier; rett før terskelverdiene virker inn, så oppdateres plotte-objektet via funksjonen *DotObjectPlotter*, se tabell 5.

Ved å ha et slikt objekt, er det lett å observere hvordan en endring av brukeren påvirker oppførselen til pådragsorganet. Endringene i dette tilfellet vil være i form av kontraksjoner, som plukkes opp av SEMG-sensorer og videre behandles i programmet. Én ting er å kunne observere hvordan pådragsorganet påvirkes kun av endringene, men å kunne se hvordan pådragsorganet endres også i forhold til plottet gir enda mer informasjon. Hvis en sammenligner oppførselen til pådragsorganet med plottet, kan en med større visshet vite hvordan en kan styre pådragsorganet for å få den ønskede oppførselen.

For å separere oppførselen, siden antallet frihetsgrader må behandles ulikt, benyttes en *switch*-blokk i *Matlab* hvor antallet frihetsgrader definerer hvert enkelt tilfelle. Dette gjelder for både genereringen og oppdateringen av plotte-objektet, siden aksesystemet ikke er likt for alle frihetsgrader. Foreløpig er det implementert støtte slik at plotte-objektet tilpasses 1-4 frihetsgrader.

Når antallet frihetsgrader overgår tre, så blir det hakket vanskeligere å håndtere plottingen i ett aksesystem. Løsningen på dette er å legge til en gradering på plotte-objektet, i dette tilfellet en fargegradering. På denne måten kan en bevege seg i fire dimensjoner, og likevel ha plottingen i ett aksesystem. Denne funksjonaliteten ble implementert ved hjelp av *Matlab*-funksjonen *scatter3*, se funksjonen *DotObjectInitialisation* i Appendiks A for å se hvordan den er initialisert.

Del III

Resultater

8 Norsk terminologi

Et av produktene fra masteroppgaven er den norske terminologien som er foreslått. Dette er begreper utviklet i samarbeid med Kristian Berrum, Anders Fougner og Øyvind Stavdahl. Alle foreslåtte begreper står i tabell 4.

9 Bidraget av intensjonsestimering

Resultatene fra denne masteroppgaven, er i stor grad bare en kortere oppsummering fra metode-delen, da spesielt fra implementasjonen i seksjon 7 i del II. Dermed får en lettere oversikt over det som er produsert i oppgaven.

Hovedvekten av implementasjonen har vært intensjonsestimeringen i figur 3. Resultatene vil bli lagt frem som i modellen av proporsjonalstyring i samme figur.

Alle funksjoner som er implementert i denne masteroppgaven, står opplistet og kort beskrevet i tabell 5. Funksjoner som bare er gjort små endringer i, står listet opp i tabell 6.

Selve bidraget av intensjonsestimeringen, ligger på wiki-sidene (Bertnum, 2013, BioPatRec web documentation) hos Ortiz-Catalan (n.d.), men i Appendiks B ligger et utkast av dokumentasjonen. En viktig ting å påpeke vedrørende dokumentasjonen i vedlegget, er at det er et arbeid i utvikling. Derfor vil ikke nødvendigvis dokumentasjonen i denne masteroppgaven, stemme overens med det hos Ortiz-Catalan (n.d.).

I tillegg til følgende underseksjoner med funksjonene bak proporsjonalstyringen, er det også et grensesnitt som er implementert. Selve utformingen av grensesnittet, er en videreutvikling av den i Linnerud (2012), og kan observeres i figur 10 i seksjon 7.2.

9.1 Lineær mapping

Resultatet av "lineær mapping"-implementasjonen, er fordelt i flere funksjoner. Med systemet på formen $F = Ax$, skjer forhåndsberegningene av A-matrisen(-e), når F og x er kjent, i funksjonen *OfflineProportionalControl*, se tabell 6. Den videre beregningen, når F er ukjent, skjer i funksjonen *LinearMapping*, se tabell 5.

Det er to estimator implementert i denne masteroppgaven for lineær mapping: normal lineær estimering og dekoblet lineær estimering. Algoritmene er beskrevet seksjon 3.2.2, og selve implementasjonen står i seksjon 7.2.2.

9.2 Ulineær støydemping

For denne implementasjonen er det laget et filter på formen i figur 12. Hensikten er å filtrere bort kjappe endringer i pådrag, slik at protesen opererer mer stabilt ved å holde konstant hastighet. Selve implementasjonen av dette filteret ligger i funksjonen *NonlinearFlutterRejectionFilter*, og står opplistet i tabell 5.

9.3 Justering av forsterkning og terskelverdier

Ved å justere parametrene i figur 10, kan en endre på forsterkningen og terskelverdiene for protesestyringen. Dette endrer oppførselen til protesen i form av mer/mindre pådrag, eller større/mindre operasjonsområder for én type bevegelse. En bevegelse vil da være enten åpning/lukking av hånden, pronasjon/supinasjon av underarmen, eller en simultan styring mellom de ulike frihetsgradene. I dette tilfellet vil det bare være for to frihetsgrader, men justeringen av forsterkning og terskelverdier støtter flere frihetsgrader. Implementasjonen av dette finnes i funksjonen *GainThresholdAdjustments*, listet opp i tabell 5.

9.3.1 Generering og oppdatering av plotte-objekt

For å illustrere behandlet data i sanntid, genereres et plotte-objekt i grafen til proporsjonalstyrings-GUIen. Genereringen skjer før *timer*-objektet initialiseres og startes, og selve oppdateringen skjer inne i justeringen av forsterkning og terskelverdier, rett før terskelverdiene får en innvirkning.

Implementasjonen av plotte-objektet har støtte for 1-4 frihetsgrader. Det er også enkelt å legge til støtte for flere frihetsgrader, dersom en vil benytte seg av samme plotte-metode. Med lik plotte-metode siktes det til å simulere alt i ett plott.

Begrep = *original ord*	**Beskrivelse**	**Eksempler**
Intensjonsestimering = *Intent interpretation*	Tolkning av målesignaler fra en bruker for å estimere brukerens intensjon. Denne typen estimering kan gjerne deles opp i to deler, se figur 3.	PatRec-algoritmer
Ulineær støydemping = *Nonlinear flutter rejection filter*	Et filter for å dempe raske variasjoner i pådraget. Hensikten er å holde pådraget til pådragsorganet stabilt.	Figur 12
Motorfunksjon = *Motor function*	Den naturlige måten å betegne styringen av muskler og funksjoner i mennesker. Modellmessig kan det ses på som om å modellere én frihetsgrad eller flere. Ved én frihetsgrad består motorfunksjonen av én type bevegelse og dens antagonist-bevegelse, mens ved flere frihetsgrader fås en mer kompleks bevegelse.	Pronasjon/ supinasjon av underarm *Kompleks bevegelse:* Åpning/lukking av hånd
Bevegelsesklasse = *Motion class*	Forskjellen mellom bevegelsesklasse og motorfunksjon, er at disse betegner en enkelt bevegelse, uavhengig av frihetsgrad. Sett i forhold til motorfunksjon, har disse bare én retning på pådraget eller null pådrag.	Pronasjon av underarm, åpning av hånd, hvile-tilstand
Brukertrening = *User training*	Trening av brukerens evne til å styre protesen. Dette vil være øvings- og erfaringsbasert trening.	
Justering av forsterkning og terskelverdier = *Gain and threshold adjustments*	Her justeres forsterkning, offset- og terskelverdier. Hensikten bak justeringene, er å få definerte operasjonssområder for hver bevegelsesklasse.	Seksjon 9.3
Styringsalgoritme = *Control algorithm*	Post-prosesserings-algoritme for å optimalisere utvelgelsen av pådrag til pådragsorganet. Målet er å redusere antallet feiltolkninger fra intensjonsestimeringen.	*Majority vote* og *Buffer output*
Beregning av signalegenskaper = *Feature extraction*	Selvforklarende navn, men se del 3 i figur 3 for plassering i protesesystemet.	*Zero-crossing* og *Waveform length*

Begrep = *original ord*	**Beskrivelse**	**Eksempler**
Av/på-klassifisering = *Crisp classification*	Klassifiseringen som gir klassene av/på-verdier. Det er mulig å bruke algoritmer for proporsjonalstyring også til av/på-klassifisering.	*KNN* og *LDA*
Pådragsorgan = *Output device*	En generalisering for enhetene som blir påsatt signaler og pådrag fra intensjons-estimeringen.	Protese, VRE

Tabell 4: Tabell med oversikt over den foreslåtte norske terminologien. Det meste av terminologien er begreper definert i tabell 1 hos (Fougner et al., 2012, s. 665), men det gis en norsk beskrivelse i denne tabellen.

Functions + *caller functions*	**Function description**
OfflineProportionalControl + *OfflinePatRecTraining*	Function to execute the Offline calculations for proportional control.
ProportionalControlTest + *OneShotPatRec*	Function to execute the testing of the proportional control. At the moment this functions only exists to make the system run, i.e. it does not do any testing.
GUI_ProportionalControl + *Load_patRec*	Function executing the GUI for the proportional control.
DotObjectInitialisation + *GUI_ProportionalControl*	Function initialising the dot object in the realtime-plot.
ProportionalControlTimer + *GUI_ProportionalControl*	Function for initiating proportional control thread.
ProportionalControlCalculation + *ProportionalControlTimer*	The input function for the ProportionalControlTimer.
LinearMapping + *ProportionalControlCalculation*	Function for executing linear mapping.
NonlinearFlutterRejectionFilter + *ProportionalControlCalculation*	Function containing a nonlinear flutter rejection filter.
GainThresholdAdjustments + *ProportionalControlCalculation*	Function executing the gain and threshold adjustments.
DotObjectPlotter + *GainThresholdAdjustments*	Function updating the dot object in the realtime-plot.

Tabell 5: Oversikt over funksjonene som er implementert i denne masteroppgaven i BioPatRec, og hvor de er kalt ifra. Koden for disse funksjonene er å finne i Appendiks A.

File-names + *functions changed* * *caller functions*	**Description of the change**
GUI_PatRec.m + *pm_SelectAlgorithm_Callback* * *GUI_BioPatRec*	Added *Proportional Control* as an alternative algorithm for system control, see *Select Algorithm*-popupmenu; in addition, two choices for estimators in the algorithm.
GUI_PatRec.fig + *pm_SelectAlgorithm* * *GUI_BioPatRec*	Added *Proportional Control* in the list of *Select Algorithm*.
OfflinePatRecTraining.m + *OfflinePatRecTraining* * *OfflinePatRec*	Added *Proportional Control* as an additional condition in the *if*-conditions for algorithm training alternatives.
OneShotPatRec.m + *OneShotPatRec* * *PatRec_SingleClassifier*	Added *Proportional Control* as an additional condition in the *if*-conditions for algorithm testing alternatives.
Load_patRec.m + *Load_patRec* * *GUI_PatRec*	Added an if-condition checking whether *Proportional Control* is chosen; and if so, it executes *GUI_ProportionalControl* instead of the alternatives.

Tabell 6: Oversikt over filene og funksjonene som er endret i BioPatRec, og hvor de er kalt ifra. Koden for disse funksjonene er å finne i Appendiks A.

Del IV

Diskusjon og konklusjon

10 Diskusjon

Norsk terminologi

Et viktig tema under denne masteroppgaven var å definere en norsk terminologi, se tabell 4. Slik kan en tydeliggjøre begrep som er oversatt fra engelsk terminologi, og som ikke eksisterer fra før av i det norske språk. Tanken bak en slik innføring er å unngå forvirring rundt begrepene (Fougner et al., 2012). En vil også å unngå opprettelsen av for mange nye begreper om samme sak. Sånn kan fokuset ligge mer på innholdet i teksten enn selve teksten rundt det.

Det neste viktige poenget for å definere en norsk terminologi, er for å fornorske oppgaveskrivingen i forbindelse med vitenskapelige artikler, som for eksempel masteroppgaven. Slik vil en beholde det norske språk som et sterkt alternativ når det kommer til ulike forskningsområder. Dette vil være viktig hvis norsk som språk skal være et alternativ i vitenskapelige artikler.

Ulempen når artiklene er på norsk, er at de mister litt av publiseringspotensialet internasjonalt. For å ta hensyn til de som ikke er norskspråklig, er det innført en del figurer på engelsk, og i tillegg lagt ut en engelsk dokumentasjon på wiki-sidene av BioPatRec-prosjektet. Koden og web-dokumentasjonen er selve produktet av bidraget i masteroppgaven til BioPatRec, og derfor det som vil være av interesse for internasjonale deltagere i prosjektet.

Format-dilemma

I seksjon 7.1 presenteres et problem når det kommer til ulike format på input og output. Egentlig ønskes det å ha like format på alle inputer og outputer. Slik er det lettere å vedlikeholde programmet, og mindre arbeid å legge til nye implementasjoner. Hvis formatet på en input eller output endres i programmet, så er nye implementasjoner mer krevende ettersom de må ta hensyn til flere måter å håndtere input og output.

Dilemmaet i implementasjonen av proporsjonalstyring, er at hvis en skal håndtere input og output i samme format, kreves det at algoritmene har støtte for det. Designet på algoritmene fra ITK-programvaren tar utgangspunkt i et annet format enn det i BioPatRec. I stedet for å ta inn bevegelsesklasser

som input, slik som i BioPatRec, tar ITK-varianten av proporsjonalstyring inn motorfunksjoner som input.

På grunn av dilemmaet vedrørende ulikt format, kan det være en tanke å endre algoritmene, men dette kan potensielt kreve en del arbeid. Til senere arbeid, anbefales det å ta tak i dette problemet og analysere dette scenarioet, eventuelt designe algoritmer bedre egnet for BioPatRec. Valget om å bare endre de eksisterende algoritmene, eller å lage noen nye algoritmer, er en vurderingssak i det fremtidige arbeidet.

Et alternativ til å endre eksisterende proporsjonalstyringsalgoritmer, er å lage en oversettingsfunksjon for å løse format-problemet. Hensikten til denne funksjonen er å oversette input- eller output-format. Først oversettes inputen til det formatet som kreves i algoritmen, men deretter vil en også endre det tilbake. Ved å oversette output-formatet til standard-formatet, unngår en unødvendig arbeid i å utvide protokollen til eksisterende kommunikasjonsprotokoller. En trenger ikke å forholde seg til forskjellige kommunikasjonsmetoder for å kommunisere med ulike input-format fra intensjonsestimeringen.

Ved implementasjon av en oversettingsfunksjon, er prinsippet bare å konvertere bevegelserklasser til motorfunksjoner, eller omvendt, se tabell 4 i seksjon 8 for terminologien. Det er allerede en form for oversetting i starten av offline-treningen til proporsjonalstyring, men en trenger også en invers-variant av denne. En slik implementasjon drøftes videre senere i diskusjonen, men står også i fremtidig arbeid, se *xSets-oversetter* i seksjon 12.

Modularitet i offline-treningen

Implementasjonen av offline-treningen er ikke helt optimal med hensyn til modularitet i dens nåværende tilstand, se seksjon 7.1. Dette skyldes at den kun tar for seg et scenario der bestemte bevegelser må være med. I dette tilfellet er det åpning/lukking av hånden og pronasjon/supinasjon av underarmen. Begrunnelsen for dette valget var at en først skulle implementere funksjonaliteten slik som den var i ITK-programvaren. Hvis det ble tid, kunne en utvide funksjonaliteten til å støtte flere frihetsgrader.

Grunnen til at modulariteten i forrige avsnitt ikke er optimal, skyldes at en må konvertere fra bevegelsesklasser til motorfunksjoner. Dette er et nok et tilfelle av format-problemet, men kan løses ved å designe en optimal oversettingsfunksjon, som presentert under drøftingen av format-dilemmaet.

Ulempen med valget om å bare implementere den gamle funksjonaliteten for protesestyring, den fra ITK-programvaren, er at modulariteten fortsatt ikke er noe bedre. Det bygges bare videre på en løsning som burde endres. Under

implementasjonen var dette heldigvis et kjent problem, altså kunne en være bevisst på det ved å prøve og gjøre videre implementasjoner mer modulære og fremtidsrettede. Tanken om å implementere en dedikert oversettingsfunksjon, var ikke tilstede under implementasjonen. Den er et produkt av ettertanker rundt implementasjonen.

Essensen i en oversettingsfunksjon vil være å lage en støtte slik at en filtrerer ut en og en bevegelsesklasse med dens tilhørende antagonist-bevegelsesklasse, for eksempel pronasjon av underarmen, og derav supinasjon som dens antagonist-bevegelse. Hensikten er å gjøre *xSets*, som er inputen til treningsalgoritmene, lesbar for proporsjonalstyringen, eller andre algoritmer som bruker motorfunksjoner. Dette er beskrevet i det fremtidige arbeidet, se *xSets-oversetter* i seksjon 12.

Hvis en får lagt til en funksjon som gjør *xSets* lesbar, vil en automatisk få bedre støtte i programmet for ITK-funksjonaliteten. Det som gjenstår er å ta for seg grensesnittet til proporsjonalstyringen, siden den også må støtte flere frihetsgrader. Dette er neste tema for diskusjonen.

Grensesnittet til proporsjonalstyring

Når det kommer til grensesnittet for proporsjonalstyring, er det rom for en del forbedringer. Det kan være forbedringer i selve utformingen av det grafiske grensesnittet, eller det kan være i form av hvordan grensesnittet fungerer og behandler data.

Den største forbedringen vil være å generalisere proporsjonalstyringen til å håndtere N-antall frihetsgrader, altså at styringen fungerer med et fritt antall frihetsgrader gitt av brukeren. Dette vil være en kombinasjon av endringer i både den grafiske utformingen, samt prosessene som kjører i bakgrunnen.

Når antallet frihetsgrader overgår tre, vil det bli en utfordring å utforme det grafiske grensesnittet. Utfordringen ligger i at dimensjoner over tre ikke naturlig plottes i form av koordinatsystemer. I så fall, vil en måtte ty til graderinger av plotte-objektet. Da i form av fargeendringer på objektet, eller endringer i form, og så videre. En har altså muligheten til å lage ulike graderinger for å illustrere flere frihetsgrader, men med denne metoden blir dette mer og mer uoversiktlig med økt antall frihetsgrader.

En mulighet for å skape oversikt, når antallet frihetsgrader øker, kan være å gi brukeren mulighet til å justere hvilke frihetsgrader han vil observere i sanntidsplottet. Selv om plottet viser selektive data fra brukerens målinger, skal alle målinger i praksis prosesseres og gå videre til pådragsorganet.

Ved plotting i et tre-akset koordinatsystem, må også operasjonsområdene gjøres tre-dimensjonale. Et mulig scenario da er å lage konus-formede områder langs aksene, en sfære i midten og en fylt sirkel som plotte-objekt. Dette er på ingen måte den eneste løsningen, men et alternativ det kan gjøres på. Selve plotte-objektet har støtte for fire frihetsgrader fordi den kan gradere fargen. Derfor må bare den grafiske utformingen utvides, hvis en vil ha flere frihetsgrader plottet i samme plott.

Hvis en ikke vil plotte alt i ett aksesystem, kan en alternativt dele det opp i flere aksesystemer. Da kan en for eksempel ha flere en-aksede systemer, hvor hver frihetsgrad representeres av et slikt system. Simultanstyring kan implementeres i form av at to eller flere terskler nås på flere av de ulike frihetsgradene. Ulempen med denne metoden er at en må observere flere grafer for å holde oversikten på pådragene. Dette vil også bli mer uoversiktlig når antallet frihetsgrader øker, men vil likevel være betydelig mer oversiktlig enn den metoden som benyttes nå.

Endres støtten for i inkludere flere frihetsgrader, må en gå gjennom grensesnittet for proporsjonalstyringen å legge til denne støtten. Dette innebærer å endre den grafiske utformingen. Den må gjøres dynamisk hvis en ønsker å simulere hele protesestyringen i ett plot for flere frihetsgrader. I denne masteroppgaven er det presentert to måter for å eventuelt angripe en endring av GUIen: å simulere alle frihetsgrader i ett plott, eller å dele opp hver frihetsgrad i en-aksede systemer.

Testing av proporsjonalstyring

I BioPatRec er det normalt at PatRec-algoritmene implementeres i form av minst to funksjoner, hvorav en trener algoritmen og den andre tester den. Fordi algoritmen til proporsjonalstyring endrer på *xSets* til et datasett med motorfunksjoner, i stedet for bevegelsesklasser, krever det en litt annen fremgangsmåte i implementasjonen av test-funksjonen.

På grunn av hvordan proporsjonalstyringen fungerer, er det ikke naturlig å gi et estimat på alle bevegelsene. Hvile-tilstanden og simultan-tilstandene defineres av terskler satt inne i selve grensesnittet til proporsjonalstyring, og vil ikke være definerte utenfor dette grensesnittet. Av den grunn vil ikke estimatene være nøyaktige uansett, siden de ikke tar hensyn til verken hvile-tilstanden eller simultan-tilstandene. Hvis en skal gi et estimat burde det vært for et satt scenario av proporsjonalstyring, for eksempel med tersklene til åpning/lukking av hånden og pronasjon/supinasjon av underarmen, satt til 45 grader, altså ingen simultanstyring, og hvile-terskelen til en radius med 0. Standarden i *PatRec*-GUIen etter offline-treningen, er å vise bevegelsesklassene

som blir sendt i input til treningsalgoritmene, samt sannsynligheten for at hver klasse blir riktig klassifisert. Gitt det kunstige scenarioet for proporsjonalstyring, kan en gi et kunstig estimat. Virkeligheten bak dette sannsynlighetsestimatet er ikke helt reelt, men kan likevel gi et inntrykk på hvor godt algoritmen presterer.

En annen måte å presentere sannsynligheten for prestasjonen til proporsjonalstyring, er å presentere hvor god avbildningen av input-data opp mot output-data er i forhold til referansedataen. Dette kan gjøres i form av tall, eller graf hvis en ønsker å etterligne den metoden brukt i ITK-programvaren. Dette er beskrevet nærmere i fremtidig arbeid, se seksjon 12.

PGT og systemtrening

I utgangspunktet var det ønskelig å starte implementasjonen av PGT som systemtrening, men dette så ut til å bli en større jobb. Dilemmaet ved en implementasjon av PGT, er at en vil kommunisere med pådragsorganet, i dette tilfellet en MC-protese, og derfor må det utvides støtte for dette i BioPatRec's kommunikasjonsprotokoll. Siden dette er en del av output-delen i implementasjonen, så var ikke dette lett å utføre i denne masteroppgaven når fokuset har vært på intensjonsestimering.

En foreløpig løsning på dilemmaet i forrige avsnitt, er å benytte seg av de allerede eksisterende output-mulighetene, altså at en benytter seg av for eksempel VRE. I dette tilfellet vil en altså ha det en trenger for å implementere PGT. Hvis en velger denne metoden, vil det likevel være et problem at en må finne en kommunikasjonsmetode videre fra proporsjonalstyringen.

Hvis det utvinnes støtte for VREen, må en likevel finne på en løsning for å illustrere hvordan den skal gripe hardere. Dette ble i arbeidet av Linnerud (2012) utviklet og gjort ved å plassere en gummiball å protesehånden, slik at en med øynene kunne observere kraften som ble utført på ballen. Hvis en skal gjøre dette i VREen, må en legge til støtte for det. Dette kan kreve litt jobb dersom en ikke er kjent med funksjonaliteten rundt VREen.

For å løse kommunikasjonsproblemet videre fra proporsjonalstyringen, finnes det flere løsninger. En løsning vil være å implementere all kommunikasjon med pådragsorgan fra bunnen av, men en bedre løsning vil være å kopiere kommunikasjonen fra de eksisterende grensesnittene. Ved bare å kopiere og adaptere den kommunikasjonen som allerede eksisterer, unngår en å gjøre det på forskjellige måter. Slik skaper en modularitet i form av like kommunikasjonsmetoder.

Når PGT er implementert, vil en at det skal være mulig å utføre trening med proporsjonale data. Dette er mer intuitivt å observere når det kan illustreres av

selve protesen, siden en gjerne vil se hvordan den varierer i motorkraft/-fart. Denne dataen kan legges i en ny variabel i *struct*-tabellen som lagres i opptaksøkten. Selve metoden for å utføre dette i implementasjonen, er dokumentert i fremtidig arbeid, se *Et alternativ til xOuts* i seksjon 12.

Sanntidsmating av EMG-data

Matingen av EMG-data i sanntid, er fortsatt noe som må legges til i programvaren. Siden dette er kode utviklet i parallell med dette prosjektet, ble det ikke tid til å slå sammen disse implementasjonene. I fremtidig arbeid er det beskrevet hvor i koden denne sanntidsmatingen må legges til. Når denne matingen er lagt til, skal proporsjonalstyring kunne fungere i sanntid når outputen implementeres.

11 Konklusjon

For å unngå språkforvirring og opprettelsen av for mange begreper om samme sak, er det foreslått en norsk terminologi for å avdekke noen begreper brukt i dette forskningsområdet, se tabell 4. Målet er at dette skal standardiseres, slik at terminologibruken blir konsekvent og at den kan utvides ved videre arbeid.

I den tidligere ITK-programvaren er det en litt høy tidsfaktor for å sette seg inn i prosjektet, og derav tar det litt tid å kunne implementere ny funksjonalitet. Derfor har en undersøkt muligheten for å ta i bruk et annet prosjekt. Prosjektet som har blitt undersøkt og påbegynt kalles BioPatRec, og er et internasjonalt prosjekt som er i stadig utvikling.

Ved et internasjonalt prosjekt, ligger mye av styrken i dets evne til å distribuere alt innhold til alle som vil delta. Om det skulle være i form av opptaksdata, altså at data fra forsøkspersoner i en annen del av verden deles, eller om det skulle være algoritmene som behandler disse dataene, går helt på det samme. Alt blir distribuert, og dette reduserer behovet for å gjenta prosedyrer om igjen. En kan brukes den funksjonaliteten som en selv ønsker, og videre fokusere på sitt eget forskningsområde. Dermed senkes terskelen for proteseforskning, fordi en er ikke avhengig av å utvikle alt selv eller ha tilgang på proteseutstyr.

Ulemper funnet underveis, er at ikke alle algoritmer er like egnet for programmets metoder i å håndtere data. Av disse algoritmene inngår proporsjonalstyring, fordi denne algoritmen tar inn motorfunksjoner som input, kontra bevegelsesklasser som brukes i BioPatRec.

Fordi bevegelsesklasser, til klassifisering, og motorfunksjoner, til proporsjonalstyring, er naturlige som inputer, så ble denne løsningen beholdt. Løsningen for å gjøre dette mer modulært var å innføre en oversettelse av bevegelsesklasser til motorfunksjoner. Implementasjonen i denne masteroppgaven bærer likevel preg av ikke optimal modularitet, for den tar kun for seg et enkelt scenario av bevegelsesklasser. Derfor er dette viktig å ta tak i ved en videre implementasjon, gitt at en vil eksperimentere med flere bevegelsesklasser.

Alt i alt er dette et bidrag av intensjonsestimeringen for simultan proporsjonalstyring, fra ITK-programvaren til BioPatRec. I videre arbeid bør en ta tak i output-delen av modellen, sett i figur 3, slik at hele systemet kan kjøres på ITK-laben. Etter det bør PGT som systemtrening implementeres. Basert på programvareanalysen, er dette best å ta i samarbeid med implementasjonen av output-delen.

12 Forslag til fremtidig arbeid

***xSets*-oversetter** Hensikten til en slik funksjon er å gjøre *xSets*-lesbar for proporsjonalstyringen, og deretter lesbar for resten av programmet. I praksis betyr dette å oversette bevegelsesklasser til motorfunksjoner, og omvendt.

I oversettelsen fra bevegelsesklasse til motorfunksjon, bør oversettelsesfunksjonen kunne definere en frihetsgrad for hver bevegelsesklasse og dens tilhørende antagonist-bevegelse. Dersom en av disse klassene ikke eksisterer, kan en påsette den null i pådrag. En kan bare opprette en datavektor av null-verdier for å oppnå et slikt pådrag.

For å oversette fra motorfunksjon til bevegelsesklasse, vil en gjøre en invers-operasjon av den beskrevet i forrige avsnitt. Denne operasjonen vil likevel være mer krevende, siden proporsjonalstyringen legger til flere bevegelsesklasser i styringen. Dette skyldes at den legger til ekstra bevegelser basert på manuelle innstillinger i GUIen.

Alternativ proporsjonalstyringsalgoritme For å unngå problemet med input- eller output-format, er det et alternativ å designe en proporsjonalstyringsalgoritme mer egnet for BioPatRec. I praksis betyr dette å lage en algoritme med bevegelsesklasser som input.

Utvide støtten for frihetsgrader i GUIen til proporsjonalstyring
I *GUI_ProportionalControl* bør en vurdere å utvide støtten til flere frihetsgrader. Da har en muligheten til å eksperimentere med flere bevegelser i sanntids-proporsjonalstyringen.

Forslaget om å utvide støtten er diskutert i diskusjonen, og står beskrevet i temaet om grensesnittet for proporsjonalstyring. Denne funksjonaliteten er allerede implementert plotte-objektet-funksjonene. Dermed er det tilrettelagt for å bruke plotte-objekt-metoden videre i utvidelsen av GUIen.

PGT og systemtrening I videre arbeid med systemtrening og BioPatRec, bør en legge til støtte for PGT. Implementasjonen bør også støtte muligheten til å benytte ulike typer treningsøkter, slik kan en teste effekten av en treningsøkt på protesestyringen. I arbeidet av Bertnum (2012) er det fokusert på å observere effekten av treningsøktene på protesestyringen; videre utforskes muligheten til å utvinne bedre treningsøkter.

For en slik implementasjon, er en nødt til å kommunisere med pådragsorganet for at treningen skal være gjennomførbar, ellers vil ikke pådragsorganet kunne gi noen respons under systemtreningen. Derfor må en utvide

kommunikasjonsprotokollen til BioPatRec, så den omfatter pådragsorganene en selv ønsker å styre, i dette tilfellet en MC-protese.

En alternativ løsning vil være å implementere muligheten for systemtrening med VREen, for da er kommunikasjonsprotokollen allerede definert. Det som er viktig å påpeke her, er at maskinvaren tilgjengelig faktisk kan kjøre VREen. På sidene til BioPatRec (Ortiz-Catalan, n.d.), er det en spesifikasjon med anbefalte krav for å benytte VREen. Gitt at disse er støttet, vil VREen mer sannsynlig kjøre med en god prestasjon.

Det viktige å være oppmerksom på i en implementasjon med VRE, er at for å gjøre PGT med VRE, så er det ikke en metode per dags dato for intuitivt å trene lukking av hånden. På en protese ble dette løst ved å legge en gummiball i protesehånden, og mens hånden lukkes kunne en observere med øynene hvor mye kraft hånden utførte på ballen. Hvis en skal legge til PGT med VRE, må en utvikle en egen metode for å trene lukking av hånden med VRE, med mindre en legger til balltrening inne i VREen.

Et alternativ til *xOuts* For å implementere proporsjonale data inn i datasettene, må det implementeres et alternativ til *xOuts*. Dette kan gjøres ved å innføre en ny variabel i *struct*-tabellen fra *RecordingSession*-delen i BioPatRec. I denne variabelen legges all proporsjonaldata fra signalopptaket.

Siden ikke alle algoritmer har støtte for proporsjonale data i algoritmetreningen, kan en bruke funksjonen *Matlab*-funksjonen *isfield*. Denne funksjonen sjekker om en variabel er til stede i en gitt *struct*-tabell. Hvis en implementerer dette i de algoritmene som støtter trening med proporsjonale data, så vil algoritmen ta inn proporsjonaldata hvis det til stede; hvis ikke velger den *xOuts* med av/på-verdier.

Dekoblet ulineær estimering I bakgrunnen nevnes tanken om å innføre en dekoblet ulineær estimering, som er basert på en tanke fra Anders Fougner. Siden dette ikke har vært fokuset i denne oppgaven, må dette drøftes og forskes videre på i et fremtidig arbeid. En kan bruke teorien om dekoblet lineær estimering som inspirasjon i denne tankeprosessen, se seksjon 3.2.2.

Blåtann-kommunikasjon med MC-protesen Hvis en legger til en blåtann-kommunikasjon i BioPatRec med MC-protesen, trenger en ikke å implementere NI-DAQen i BioPatRec for å kommunisere med protesen. En blåtann-implementasjon vil i tillegg gjøre protesestyringen mer behagelig, fordi en ikke trenger å ha en DAQ hengende på seg i testingen av MC-protesen.

Test-algoritme for proporsjonalstyring For å ha et estimat på hvor godt algoritmen fra offline-treningen presterer, bør det implementeres en test-

funksjon. I BioPatRec er dette gjort ved å gi en sannsynlighet for at hver bevegelsesklasse estimeres riktig. For proporsjonalstyring vil ikke dette gi et særlig godt estimat, siden algoritmen tar utgangspunkt i motorfunksjoner.

Forslaget for å gi en prediksjon på algoritme-prestasjonen, er å implementere graf-varianten fra ITK-programvaren. I denne grafen, plottes referansedataen på grafen, og så plottes dataen fra avbildningen i algoritmetreningen oppe på referansedataen. Slik kan brukeren se resultatet av signalopptaket, og bedømme om disse dataene er tilstrekkelige for protesestyring. I prinsippet skal dette kunne observeres ved å se hvor godt avbildningen stemmer overens med referansedataen, altså hvor like de to grafene blir.

Utvidelse av norsk terminologi I videre arbeid med BioPatRec eller protese-forskning, bør en utvide terminologien etter hvert som det innføres nye begreper. En kan gjerne finne en form for offentlig publisering for dette formålet, for eksempel ved å legge til en god oversikt på *Wikipedia*, eller andre alternativer. Slik er det lett å slå opp uklare begreper hvis en trenger en beskrivelse eller eksempler. I tillegg vil en ha en oversikt over alle begrepene som etter hvert innføres.

Sanntidsmating av EMG-data i proporsjonalstyring For at proporsjonalstyringen skal kunne fungere med sanntidsmating av EMG-data, må en legge til denne matingen i programmet. Dette legges til i *GUI_ProportionalControl*-filen i det markerte området under *pb_run_Callback*-funksjonen. I dette området må en starte økten med sanntidsmatingen av EMG-data. Når økten skal stoppes legges det til i det markerte området senere i funksjonen, i det en trykker stopp-knappen.

EMG-dataen som sendes i sanntid, oppdateres i form av en global variabel. Denne variabelen deklareres i det en starter sanntidskjøringen av proporsjonalstyring, og slettes i det den stoppes. Slik unngår en problemer med at det opprettes globale variabler som ligger i minnet. Selve oppdateringsintervallet av den globale variabelen, vil ikke ha noen effekt av å bli satt under oppdateringsintervallet av den i timer-objektet. Disse bør egentlig være like, men en må passe på så disse synkes, slik at det ikke blir noen tidsforsinkelser. Oppdateringsintervallet brukt i *timer*-objektet er foreløpig satt til 0,5 sekunder, se listing 1. Denne er bare å endre etter behov i senere arbeid.

Implementasjon av output-delen i protesestyring Den resterende delen som må implementeres i BioPatRec er output-delen, se output i figur 3. Fremgangsmåten for å implementere dette er å legge til en aktivasjonsprofil, som beskrevet i paragrafene om aktivasjonsprofil i seksjon 3.2.2. I den-

ne implementasjonen kan en legge til muligheten for å modifisere denne aktivasjonsprofilen i grensesnittet for proporsjonalstyring. Selve funksjonaliteten bak aktivasjonsprofilen står dokumentert i arbeidet av Fougner (Fougner, 2013, s. 113). Denne implementasjonen er foreslått å legge til i funksjonen *ProportionalControlCalculation* i det markerte ommrådet.

Referanser

Berrum, K. (2013), Biopatrec som forskningsplattform med fokus på signalopptak, Masteroppgave, Norges teknisk-naturvitenskapelige universitet, Trondheim, Norge.

Bertnum, A. B. (2012), Prosthesis guided training, Term project, Norwegian University of Science and Technology, Trondheim, Norway.

Bertnum, A. B. (2013), 'Biopatrec - web documentation contribution'.
URL: *https://code.google.com/p/biopatrec/wiki/NTNU*

Chicoine, C., Simon, A. and Hargrove, L. (2012), Prosthesis-guided training of pattern recognition-controlled myoelectric prosthesis, *in* 'Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE', IEEE, pp. 1876–1879.

Duda, R. O., Hart, P. E. and Stork, D. G. (2001), *Pattern Classification*, 2 edn, John Wiley & Sons.

Fougner, A. (2013), Robust, Coordinated and Proportional Myoelectric Control of Upper-Limb Prostheses, PhD thesis, Norwegian University of Science and Technology.

Fougner, A., Stavdahl, Ø. and Kyberd, P. J. (submitted), 'Simultaneous proportional control of dualfunction upper limb prostheses', *IEEE Transactions on Biomedical Engineering* .

Fougner, A., Stavdahl, Ø., Kyberd, P. J., Losier, Y. G. and Parker, P. A. (2012), 'Control of upper limb prostheses: Terminology and proportional myoelectric control - a review', *IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING* **20**(5), 663–677.

Linnerud, Å. S. (2012), Lab-oppsett for proteseforskning, Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Lock, B. A., Simon, A. M., Stubblefield, K. and Hargrove, L. J. (2011), Prosthesis-guided training for practical use of pattern recognition control of prostheses, *in* U. o. N. B. Institute of Biomedical Engineering, ed., 'MEC '11 Raising the Standard - University of New Brunswick's International Conference on Advanced Limb Prosthetics', Symposium Proceedings, pp. 61–64.

Ortiz-Catalan, M., Brånemark, R. and Håkansson, B. (2013), 'Biopatrec: A modular research platform for the control of artificial limbs based on pattern recognition algorithms', *Source code for biology and medicine* **8**(1), 11.

Ortiz-Catalan, M. J. (n.d.), 'Biopatrec'.
URL: *http://code.google.com/p/biopatrec*

Rybkin, V. A., Soldatenko, I. S. and Bandurina, T. V. (2005), On the technique of linguistic classification based on fuzzy neural network, *in* E. Montseny and P. Sobrevilla, eds, 'EUSFLAT/LFA 2005 Proceedings', Symposium Proceedings, pp. 512–517.

Simon, A. M., Lock, B. A., Stubblefield, K. and Hargrove, L. J. (2011), Prosthesis-guided training increases functional wear time and improves tolerance to malfunctioning inputs of pattern recognitioncontrolled prostheses, *in* U. o. N. B. Institute of Biomedical Engineering, ed., 'MEC '11 Raising the Standard - University of New Brunswick's International Conference on Advanced Limb Prosthetics', Symposium Proceedings, pp. 65–68.

Appendiks A CD

Innholdslisten til CDen:

A.1 Masteroppgaven er her vedlagt som en enkel pdf.

A.2 Referanser tilgjengelig i pdf-format ligger i `/Referanser/`.

A.3 BioPatRec som fullstendig programvare med implementasjoner ligger vedlagt i `/BioPatRec/BioPatRec_TVÅ`.

I mappen `/BioPatRec/BioPatRec_TVÅ/PatRec/Proportional Control/` ligger bidraget fra denne masteroppgaven til intensjonsestimeringen for proporsjonalstyring i BioPatRec.

A.4 LaTeX-prosjektet og figurer ligger vedlagt i `/LaTeX-prosjekt/`, hvor figurene ligger i `/LaTeX-prosjekt/gfx/`. Alle figurene er tilgjengelige som enten pdf-, eps- , png- og/eller jpg-formater.

Appendiks B Web-dokumentasjon

I dette vedlegget følger et utkast av web-dokumentasjonen lagt ut på wiki-sidene til BioPatRec (Ortiz-Catalan, n.d.). Linken direkte til sidene, er å finne Bertnums bidrag til BioPatRec-sidene (Bertnum, 2013, BioPatRec web documentation).

NTNU

A description of the NTNU contribution

Updated Today (82 minutes ago) by bert...@gmail.

NOTE: This is an ongoing development and therefore the documentation is not yet finalised.

Introduction

These pages will contain a documentation of the contributions from NTNU. Since this still is a ongoing work, it will be separated from the main hierarchy at this moment.

At this instance, the purpose of this work, is to implement the functionality already in place at the lab at the Institute of Cybernetics at NTNU, over to the BioPatRec system.

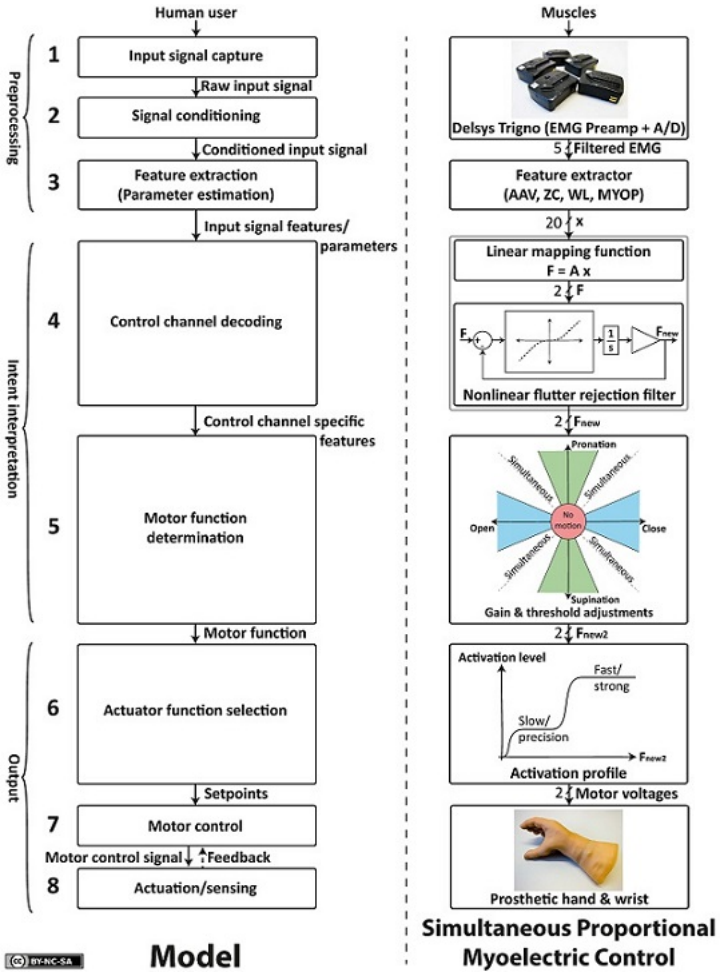
The main area of development the last years, have been simultaneous proportional myoelectric control [1].

Details

Currently, the model being developed, is split into four parts:

- Preprocessing
- Intent interpretation
- Output
- System training
  - Screen guided training (SGT)
  - Prosthesis guided training (PGT)

Each of the three first parts, are representing the steps in the model from the work of Fougner [1], and beside the model is a scenario of simultaneous proportional control, see figure under this paragraph. The last part, system training, is containing information about the training of the system, thereof SGT and PGT. In the figure being shown, the system training is fed into part 4 during the offline-training of the algorithms.



References

1. Fougner, Anders. Robust, Coordinated and Proportional Myoelectric Control of Upper-Limb Prostheses. Diss. Norwegian University of Science and Technology, 2013.

NTNU_Intent_interpretation

A description about the intent interpretation of the NTNU contribution.

Updated Today (2 minutes ago) by bert...@gmail.

NOTE: This is an ongoing development and therefore the documentation is not yet finalised.

Intent interpretation

Simultaneous proportional control

As seen on the front page of the NTNU contribution, there is figure showing the details for this part. It is divided into two parts, containing three operations for simultaneous proportional control:

- Linear mapping
- Nonlinear flutter rejection filter
- Gain and threshold adjustments

These operations together make the intent interpretation for the simultaneous proportional control.

NTNU_Linear_mapping

The linear mapping function of the simultaneous proportional control

Updated Today (70 minutes ago) by bert...@gmail.

NOTE: This is an ongoing development and therefore the documentation is not yet finalised.

Linear mapping ¶

Given a system on the form of equation 1 [Eq. 1], and that there is linear mapping, there are estimators designed to create the relation between the input and the output of the system [1] (norwegian source). Some equations are wrong in the first paper [1], but corrected in the paper of Bertnum [2].

$$F = Ax \quad \textbf{[Eq. 1]}$$

In the formula, x is the input feature vector, A is the matrix mapping x to F, and F is the output vector containing the output intent estimate for the prosthesis control.

Two steps are necessary for the estimators in linear mapping. Firstly, the algorithm must go through the offline-training in BioPatRec. During the offline-training, the F and x is known from the system training, being in the xSets data structure. Thus A can be calculated and used for further estimation. Lastly, the algorithm is ready to be run, and can do so in accordance to the estimator chosen. Both of these steps have a choice of estimators:

- Normal linear estimator
- Decoupled linear estimator

The offline-training is implemented within the *OfflineProportionalControl* function, and the online-training is implemented within the *LinearMapping* function. Both are inside the proportional control folder.

Normal linear estimator

This estimator uses linear regression for its calculation of the matrix A. Which in Matlab can be interpreted as in equation 2 [Eq. 2].

$$A = F/x \quad \textbf{[Eq. 2]}$$

Here it's important to notice, that the Matlab-operator "/", is interpreted according to the format of the input. See documentation within Matlab for details ("doc /" or "doc mrdivide").

After estimating the A matrix, the algorithm is ready for realtime-calculation. This calculation is done through equation 1 [Eq. 1].

Decoupled linear estimator

In this estimation the F vector is decomposed to three feature vectors: mean [Eq. 3], standard deviation [Eq. 4] and form [Eq. 5].

$$F_{mean} = mean(F) \quad \textbf{[Eq. 3]}$$

$$F_{std} = std(F) \quad \textbf{[Eq. 4]}$$

$$F_{form} = (F-F_{mean})/F_{std} \quad \textbf{[Eq. 5]}$$

For the estimation of the A matrix in decoupled linear estimation, equation 6 [Eq. 6], 7 [Eq. 7] and 8 [Eq. 8] are used.

$$A_{mean} = F_{mean}/x \quad \textbf{[Eq. 6]}$$

$$A_{std} = F_{std}/x \quad \textbf{[Eq. 7]}$$

$$A_{form} = F_{form}/x \quad \textbf{[Eq. 8]}$$

After the offline-calculation, the A matrix can be run through equation 9 [Eq. 9] with the realtime-feed of EMG-data.

$$F = (A_{form}\,x).*(A_{std}\,x) + A_{mean}\,x \quad \textbf{[Eq. 9]}$$

References

1. Linnerud, Ådne Solhaug. Lab-oppsett for proteseforskning. Diss. Norwegian University of Science and Technology, 2012.
2. Bertnum, Alf Bjørnar. BioPatRec som forskningsplattform med fokus på intensjonsestimering og systemtrening. Diss. Norwegian University of Science and Technology, 2012.

NTNU_Nonlinear_flutter_rejection_filter

Information about the nonlinear flutter rejection filter

Updated Today (62 minutes ago) by bert...@gmail.

NOTE: This is an ongoing development and therefore the documentation is not yet finalised.

Nonlinear flutter rejection filter

The purpose of this filter is to suppress fast and small-amplitude variations in the estimate of the output; thus, the estimate is smoothed and the motors of the output-device are relieved of oscillating actuations. This improves the stability of the prosthesis control, as the actuation estimates are kept constant for small variations in the estimate of the input to the motors [1].

Details

For details around the filter, see the nonlinear flutter rejection filter in the figure on the front page of the NTNU contribution. The only thing to notice, is that the nonlinearity in the filter is defined by $y = |x| \tanh(k\,x)$.

The code is implemented within the function called *NonlinearFlutterRejectionFilter*, and is located inside the proportional control folder.

References

1. Fougner, Anders. Robust, Coordinated and Proportional Myoelectric Control of Upper-Limb Prostheses. Diss. Norwegian University of Science and Technology, 2013.

NTNU_Gain_and_threshold_adjustments

Information about the gain and threshold adjustments

Updated Today (11 minutes ago) by bert...@gmail.

NOTE: This is an ongoing development and therefore the documentation is not yet finalised.

Gain and threshold adjustments

The purpose of the gain and threshold adjustments are to define a domain for each motion class [1]. Thus, it will be possible to set specific actuations to the output-part. In addition, it applies the offset and gain values, set in the GUI, to the acutation estimate. The offset and gain values happens first, then the threshold adjustments are applied and a new actuation estimate is calculated.

Details

2D scenario

The working spaces defined, can be observed in part five of the proportional control figure on the front page.

The first steps in this function, are to apply the offset and gain adjustments. Then the threshold adjustments are applied, which calculates the new actuation estimate.

Principle of the threshold adjustments (code implementation is more optimised):

```
actuationEstimate = [x; y];


if (actuationEstimate(1) and actuationEstimate(2)) are within red inner circle
 actuationEstimate(:) == 0;

elseif actuationEstimate[x;y] is within x-axis area (blue)
 actuationEstimate(2) = 0;

elseif actuationEstimate[x;y] is within y-axis area (green)
 actuationEstimate(1) = 0;

elseif actuationEstimate[x;y] is within simultaneous area (white)
 actuationEstimate is decomposed to the the middle axis
 actuationEstimate(:) = midlleAxisValue;

end
```

References

1. Fougner, Anders, et al. "Control of Upper Limb Prostheses: Terminology and Proportional Myoelectric Control-A Review." (2012): 1-1.
2. Fougner, Anders. Robust, Coordinated and Proportional Myoelectric Control of Upper-Limb Prostheses. Diss. Norwegian University of Science and Technology, 2013.