Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Informasjonsteknologi Automatisering og signalbehandling	Vår.....semesteret, 2018. Åpen
Forfatter: Hartvik Line	..... (signatur forfatter)

Fagansvarlig:
Karl Skretting
Veileder(e):
Karl Skretting

Tittel på masteroppgaven:

Optimalisering av gangmønster for firebeint robot med en genetisk algoritme

Engelsk tittel:

Optimizing locomotion control for quadruped robot using a genetic algorithm

Studiepoeng:
30

Emneord: Robot Maskinlæring Genetiske algoritmer	Sidetall: ....49.... + vedlegg/annet: ....1.... Stavanger, ...15/6-2018... dato/år

Forord

Denne masteroppgaven er skrevet som avsluttende hovedoppgave under mitt studie ved Institutt for data- og elektroteknologi ved Universitetet i Stavanger.

Jeg vil først takke Karl Skretting for god veiledning og tilgjengelighet under arbeidet med masteroppgaven.

Videre vil jeg takke Instituttet for data og elektroteknikk for økonomisk støtte for å produsere robotplattformen. Takk også til deres ansatte for motivasjon og oppfordring til å skrive oppgave innen maskinlæring med praktisk og kreativ vinkling.

Sammendrag

Virveldyr har igjennom evolusjon utviklet kompliserte sentrale mønstergeneratorer (CPG) i sentralnervesystemet for lokomosjon for å avlaste hjernen med sykliske arbeidsoppgaver. Innen robotikk er det interessant å etterligne disse mønstergeneratorene for bevegelseskontroll. Det er laget en forenklet modell for å etterligne CPG med Hopf-stabile oscillatorer i et periodisk nevralnettverk. Noder i nevralnettverket optimaliseres ved hjelp av en evolusjonsalgoritme kalt genetisk algoritme. Det er konstruert en firebeinet "quadroped" robot hvor nevralnettverket er implementert. Videre er det laget et treningssystem som trener opp roboten basert på ulike ytelsesmål som hastighet, balanse og energieffektivitet.

Resultatene viser at treningssystemet basert på genetiske algoritmer har en unik evne til å raskt finne løsninger i et komplekst løsningsrom med bedre resultat enn ved typisk "ovenfra og ned design". Etter trening sammenlignes egenskaper innen lokomosjon med manuelt instilte parametere. De opptrente parameterene gir roboten en signifikant forbedring iht. stabilitet og energieffektivitet.

Det er også testet hvordan genetiske algoritmer kan anvendes til å trene opp robotens adaptive egenskaper. Ved å ødelegge to av beinene til roboten, klarte den ikke å balansere med tidligere optimale bevegelsesparametere. Den ble så overlatt til treningssystemet, hvor roboten raskt lærte seg å finne nye måter å gå på.

Forkortelser og ordforklaringer

Forkortelse	Engelsk	Norsk beskrivelse
	Gait	Bevegelsessammenheng mellom gangmønster (ganglag)
	Swing phase	Pendelfase i bevegelsesmønster (fotløft)
	Stance phase	Standfase i bevegelsesmønster (bakkekontakt)
	Autonomos robot	Robot som tar valg basert på omgivelser og prosess i større grad
	Bioroboticks	Fagområde som kombinerer robotikk og biologi
	Morfology	Kroppsbygningen til en organisme
	Quadruped	Mobil robotplattform med fire bein
	Fitness function	Ytelsesfunksjon
AI	Artificial intelligence	Kunstig intelligens
COM	Communication Port	Seriell kommunikasjonsport
COT	Cost of transport	Energikost for transport
GA	genetic algorithms	Genetiske algoritmer
IMU	Inertial measurement unit	Treghetssensor
RNN	Recurrent Neural Network	Periodisk neuralnettverk
CAD	Computer Aided Design	Dataassistert konstruksjon

Innhold

Kapittel 1

Innledning

Roboter og automatiserte systemer blir en stadig større del av hverdagen innen mange fagområder. Økonomiske insentiver driver en massiv omstrukturering innen automatisering av systemer og arbeidsoppgaver. Utviklingen skjer fort, men årsaker som beskrevet av Moravec paradoks[1] har holdt igjen utviklingen av fysiske roboter sammenlignet med annen digital teknologi. Grad av autonomi og evnen til å tilpasse seg ukjente omgivelser er store utfordringer innen mobile robotplattformer. I tillegg er det resurskrevende å utvikle og optimalisere systemer til roboter som skal samhandle med den fysiske omverdenen med typisk "ovenfra og ned" design. Ingeniører må forhåndsprogrammere mulige tenkte scenarier slik at roboten kan ta riktige valg når den kommer over problemstillinger den ikke er kjent med.

Denne oppgaven har til hensikt å utprøve selvlæring innen roboter, og da spesielt hvordan maskinlæring og genetiske algoritmer (GA) kan anvendes til opptrening av lokomosjon til en fysisk robot. Ved å gi roboter en viss grad av selvinnsikt for evaluering av egen ytelse, kan roboter få frihet til å utvikle seg selv. Vi som utviklere definerer rammene roboten kan arbeide i og hvilke egenskaper og ytelser vi ønsker av roboten.

1.1 Mobile roboter

En robot er en menneskelaget maskin som utfører arbeidsoppgaver. Noen arbeidsoppgaver ble opprinnelig utført av mennesker. Andre arbeidsoppgaver har blitt muliggjort etter at roboten ble introdusert. Oppgavene kan utføres automatisk eller ved fjernstyring. Navnet "robot" er først introdusert av Karel Capek i 1923 hvor han beskriver menneskeskapte maskiner i et teaterstykke. Ordet robot kommer fra det tsjekkiske ordet "robota" som betyr "tvunget arbeid".

Implementering av roboter i arbeidsoppgaver kan både redusere kostnader samt øke grad av nøyaktighet, produksjonshastighet og utholdenhet. Anvendelse av roboter kan og redusere risiko under farlige arbeidsoppgaver, som for eksempel lakkering, minerydding, dykking eller brann og redningsarbeid.

Ved å gi en robot evnen til å forflytte seg, kan man utvide horisonten for applikasjoner hvor roboter kan anvendes. Innen utvikling av mobile roboter er det mange problemstillinger det arbeides med å forbedre:

[1] Moravecs paradoks: Høy-nivå beslutningsprosess krever ikke mye beregningskraft, mens lav-nivå beregninger, som bruk av sensorer for å sanse omverdenen og manøvrering krever enorm grad av beregningskraft og kompleksitet

Lokalisering

Mobile roboter må ha en måte å beregne sin posisjon. Mange systemer kan kombineres for at en robot skal kjenne sin lokasjon som f.eks. GNSS-system, IMU-sensor, lidar/radar teknologi og kamera.

I tilleg bør en mobil robot ha systemer for å gjenkjenne objekter i nærheten. Dette for å unngå kollisjoner og planlegge rute for å løse arbeidsoppgaver.

Energiforsyning

Ofte er mobile roboter drevet at elektrisk energi. Dermed er de også begrenset til energien den kan bære med en gitt batterikapasitet. For å lade opp batteriene kan f.eks solenergi høstes med solcellepaneler, eller roboten kan manøvrere seg til en strømforsyning.

Grad av autonomi

For mobile roboter trengs en høy grad av autonomi. En mobil robot må hele tiden overvåke sine omgivelser, og ta valg basert på hvordan den skal samhandle med omverden. Roboten må unngå kollisjoner, planlegge transport og løse oppdragene de er gitt. Autonome systemer bør utvikles slik at de kan ta avgjørelse for å ikke gå i lås når uforutsette eller ukjente hendelser oppstår. Et eksempel på et autonomt styringssystem er "hybrid deliberate/reactive paradigm".

Maskinlæring av typen som er foreslått i denne oppgaven kan være en byggeblokk i et autonomt system. Dermed kan roboten selv trene seg opp til en fungerende løsning når den innser at den står ovenfor et problem.

Lokomosjon

En mobil robotplatform må ha en mekanisme som gir forflytning. Det kan være ved hjelp av hjul, belter, bein, vinger eller oppdrift. Ulike arbeidsoppgaver gir ulike behov for metoder.

Fordelen med roboter som anvender bein til lokomosjon er den unike evnen til å manøvrere seg i terreng og over hinder. Det er også ansett som mye tryggere enn flyvende roboter, spesielt dersom roboten skal samhandle med mennesker.

1.1.1 Energikostnad for lokomosjon

Lokomosjon, evnen til å bevege seg fra et sted til et annet, har vært en essensiell brikke i et evolusjonsperspektiv til det mangfoldet av arter vi finner rundt oss i dag. I millioner av år har naturen prøvet og feilet seg fram til et unikt mangfold av arter. Hver art ofte med en unike egenskaper for å overleve og samspille ovenfor andre arter i sine territorielle områder. Så når ingeniører skal designe og utvikle energieffektive robotplattformer, hva er vel bedre en å hente inspirasjon fra naturen?

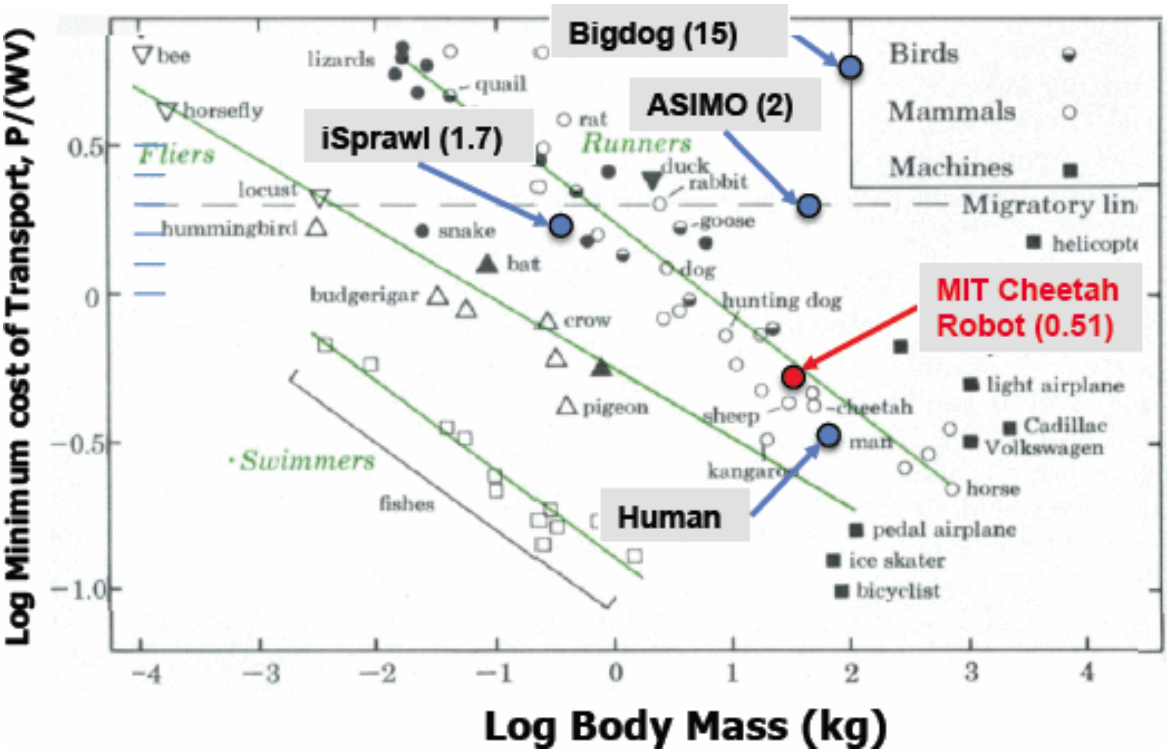
For å beregne effektivitet under lokomosjon har V. A. Tucker (1975)[1] definert et felles mål på energieffektivitet under transport. Tucker definerer utrykket "cost of transportation", COT, som representerer hvor mye energi et dyr forbruker per vektenhet ganger avstand tilbakelagt.

$$COT = \frac{E}{mgd} \qquad \text{[Enhetsløs]} \quad (1.1)$$

Hvor:

E = energi forbrukt (Joule) $[\frac{kg}{m^2 s^2}]$
m = masse [kg]
g = tyngdeakselerasjon $[\frac{m}{s^2}]$
d = horisontal distanse avlagt, [m]

Tucker har undersøkt energieffektiviteten til forskjellige dyrearter, oppsummert i figur 1.1. Sangok Seok et. al, 2013[20] har lagt til datapunkter fra forskjellige roboter som anvender bein som lokomosjon til sammenligning.



Figur 1.1: Sangok Seo et. al,2013 page 3307 [20]

Det er spesielt interessant hvordan arters effektivitet har et lineært forhold mellom arters transportkostnad og vekt. Det er også interessant hvordan svømmere er mer effektive enn flygere, som igjen er mer effektive en arter med bein. Altså er lokomosjon ved hjelp av bein lite effektivt, men evolusjonen har allikevel utviklet arter for å vandre og utforske jordens store landområder.

Lav COT verdi sees på som en god egenskap for lokomosjon, og kan dermedanvendes som ytelsesmål for opptrening av roboten. I tillegg kan robotens ytelse sammenlignes med andre dyr og roboter i samme vektklasse.

I en robotapplikasjon er det interessant å vite hvor mye av energiforbruket som går med til faktisk forflytning. Sangok Seok et. al, 2013[20] har analysert energiforbruket for deres firebeint robot Cheetah, og funnet at kun 26,5% av energiforbruket går til mekanisk kraft for lokomosjon. 6,5 % går med for å drive elektronikk og 67% er tapt til varme. Det er altså fortsatt en del å hente innen energieffektivitet i firebeint-roboter.

1.2 Biologi som inspirasjon

Fagfeltet hvor roboter henter inspirasjon fra biologi kalles biorobotikk. Roboten som er konstruert i denne oppgaven har flere elementer som er inspirert av biologi.

1.2.1 Robotkonstruksjon med inspirasjon fra en katt

Martin Fischer og Reinhard Blickhan 2006[6] har studert hvordan anatomien til virveldyr har utviklet seg igjennom evolusjon. De har funnet at dyr som har utviklet seg fra 2-segment til 3-segment beinkonstruksjon har en stor fordel innen stabilitet, hastighet og energieffektivitet. A. Sproewitz et. al, 2009[21] foreslår et 3-segment design til beinene for firbeinte robotapplikasjoner. I arbeidet med denne oppgaven er det bygget en firebeinet-robot med bein basert på dette 3-segment-designet.

1.2.2 Sentrale mønstergeneratorer

I ryggraden til alle virveldyr finnes nervesammenkoblinger som styrer muskelaktuatorer[14]. Disse sammenkoblingene produserer periodiske signaler som kan trenes opp til ulike mønster. Nervesammenkoblingene kalles sentrale mønstergeneratorer (CPG, central pattern generators). Mønstergeneratorene har til hensikt å avlaste hjernen i å utføre repetitive og syklsike motoriske oppgaver, som for eksempel pust, tygging eller lokomosjon.

Ved hjelp av disse mønstergeneratorene kan et dyr opprettholde et bevegelsesmønster til muskler som styrer lokomosjon uten konstant korrigering fra hjernen. Hjernen fungerer da som et overordnet kontrollorgan som kun sender signaler som ønsket hastighet og korrigeringer for balanse.

I artikkelen "Central pattern generators for locomotion control in animals and robots: A review"[9] oppsummerer Auke Ijspeert hvordan CPG kan representeres med en matematisk modell for anvendelse i robotapplikasjoner.

Roboten produsert i denne oppgaven har implementert en modell av mønstergeneratorer som kun trenger overordnede styresignaler for å iverksette et lokomosjonsmønster. Videre er disse oscillatorene implementert i et kunstig nevralnettverk som trenes opp ved hjelp av genetiske algoritmer.

1.2.3 Evolusjonsalgoritmer

I naturen kan man si at biologiske systemer utvikles mot en optimal struktur hvor arter lever i et balansert samspill. Hver art utvikler sin populasjon, ut fra evnen til å overleve og evnen til reproduksjon. Suksessraten til hvert individ er gitt av et komplekst forhold basert på hvor godt organismen klarer å overleve i sitt miljø ("Økologisk utvalg") og evnen til reproduksjon ("Seksuelt utvalg"). Stokastiske forhold spiller også inn.

Fagfeltet som anvender algoritmer basert på evolusjon for maskinlæring kalles "Evolutionary robotics". Denne oppgaven anvender en evolusjonsalgoritme kalt genetisk algoritme (GA) for å optimalisere gangmønsteret til roboten. Egenskaper som beskriver robotens bevegelse blir lagt inn vektorer tilsvarende kromosomene til DNA-molekyl. Algoritmen utvikler denne vektoren iterativt mot en bedre suksessverdi, gitt en ytelsesfunksjon. Underveis i treningsalgoritmen gjøres det ytelsesmål av roboten som avgjør om kombinasjonen av gener vil overleve og utvikles til neste generasjon.

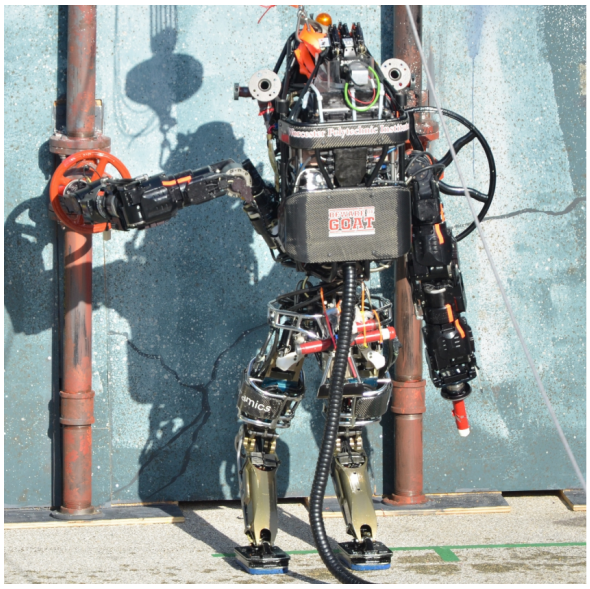
På denne måten kan vi som utviklere definere to ting: Spillerommet roboten har å prøve og feile i, samt hvilke egenskaper vi ønsker roboten skal utvikle seg til å bli bedre i. Så er det opp til roboten å utvikle seg mot en god løsning selv.

1.3 Relatert arbeid

Dette delkapittelet gir noen interessante eksempler på relevant forskning innen mobile robotplattformer.

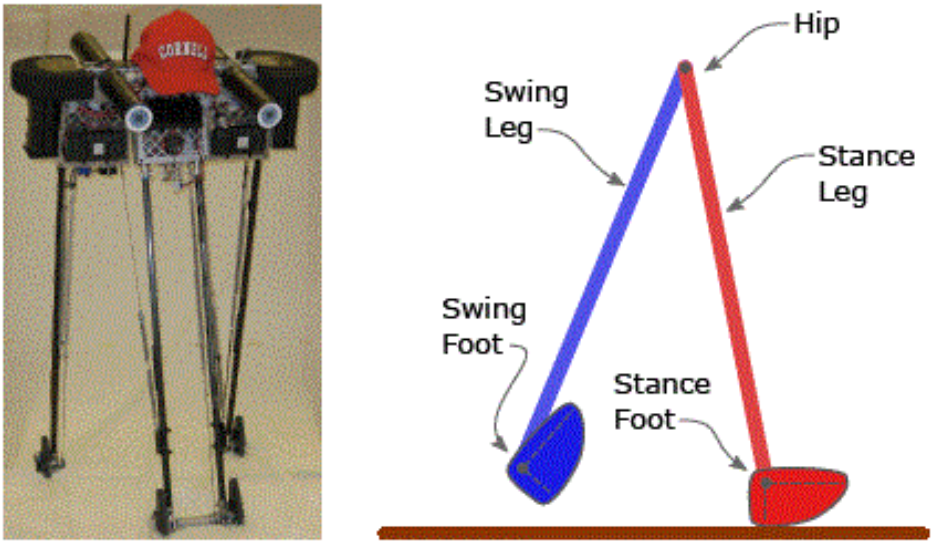
1.3.1 Mobile robotplattformer

Boston Dynamics kan sees på som en av de ledende innen robuste mobile robotplattformer. De har utviklet flere roboter med eksepsjonell manøvreringsevne og høy grad av autonomi. Siyuan Feng et al.[5] beskriver hvordan inverskinematikk og høynivåkontroller er utviklet for å håndtere et skadested i en krisesituasjon i humaniodroboten Atlas.



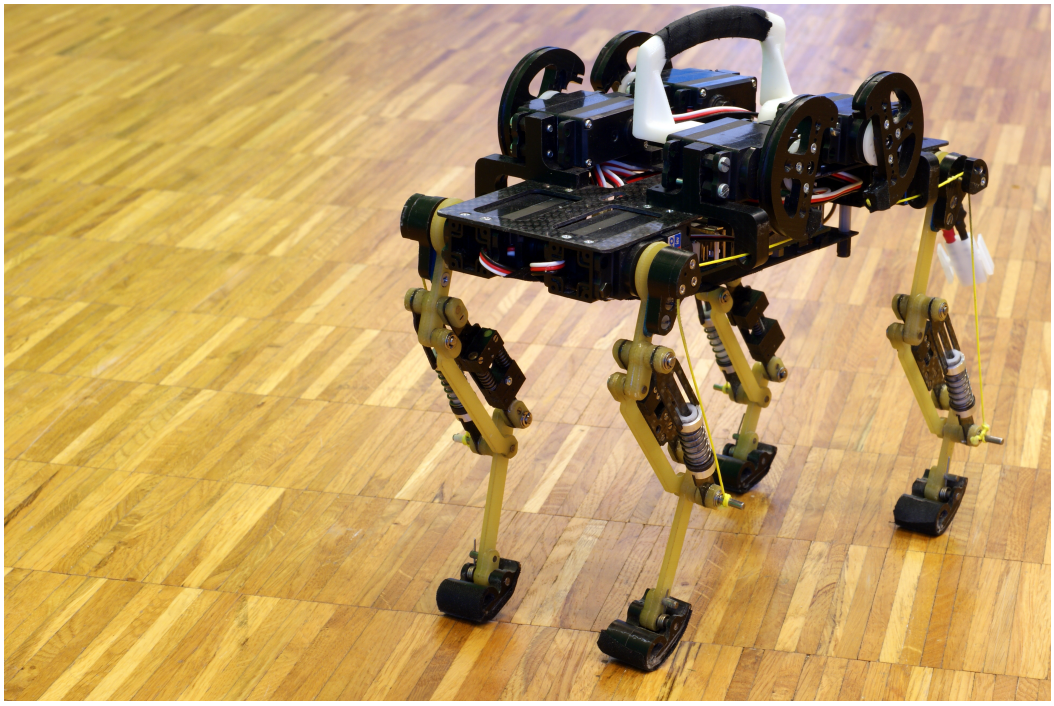
Figur 1.2: Roboten Atlas under konkurransen DARPA Robotics Challenge Finals
©[IEEE-RAS 15th International Conference on Humanoid Robots 2015][5]

Ved Cornell University i USA er det laget en særdeles effektiv robot. Roboten Ranger1.3 har rekorden for mest energieffektive firebeint robotplattform (pr. juni 2018). Ranger har en COT verdi på 0,19, og har løpt ultra-marathon på 65.12 km uten lading eller menneskelig hjelp.



Figur 1.3: Roboten Ranger
©[2016 IEEE International Conference on Robotics and Automation (ICRA)][11]

Roboten som er produsert i denne oppgaven er inspirert av roboter utviklet ved EPFL Biorobotic Labratory i Sveits. Ved EPFL har forskerene stort fokus på biorobotikk. De har mange interessante prosjekter, f.eks gjenskaper Peter Eckert et al.[4] sentrale mønstergeneratorer (CPG) for anvendelse i roboter, både for styring av beinene og en fleksibel kontrollerbar ryggrad. Robotenes mekaniske konstruksjon og bevegelsesmønster er etterlignet morfologien til firbeinte dyr eller amfibier. De utvikler dynamisk stabile roboter som ikke trenger lukkede reguleringssløyfer med sensortilbakemelding for stabilitet. Bilde 1.4 viser deres robot CheetahCub, som har en toppfart på 6,9 kroppslengder per sekund. I 2013 var dette raskest i verden for 4 beint roboter under 30 kilo.



Figur 1.4: Roboten CheetahCub. tre-sementert beinkonstruksjon
Copyright: Biorobotics Laboratory, EPFL

1.3.2 Evolusjonsalgoritmer

Dette delkapittelet viser noen eksempler på arbeid andre har utført innen evolusjonsalgoritmer.

Simulering

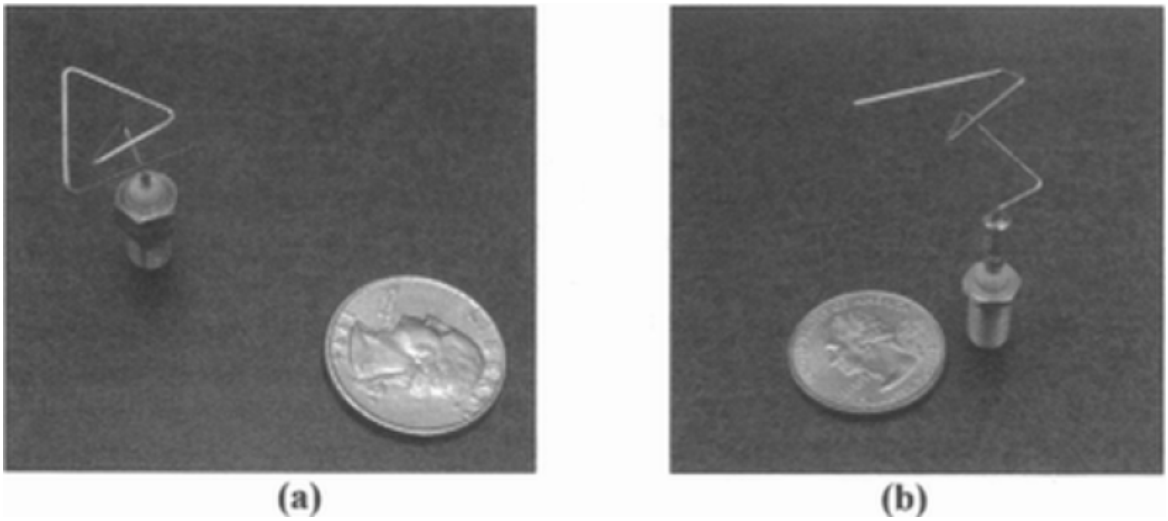
Karl Sims eksperimenterte allerede i 1994 med hvordan GA kan utvikle datasimulerte skapninger i et kunstig 3-dimensjonalt rom[3]. Skapningene ble programert til å følge gitte fysiske regler i et programmert miljø hvor skapningenes morfologi utvikles gjennom generasjoner. Ytelsesmål ble så rangert basert på om skapningene nådde resursene først. Forskjellige arter utviklet seg mot å løpe, krype eller hoppe mot resursene. Men så utviklet andre arter seg til å heller ta ut konkurrentene sine. Dette er et eksempel på hvordan roboter programert med evolusjonsalgoritmer kan ende opp med løsninger som ingeniørene ikke har forestilt seg på forhånd.

Fysiske roboter

Evolusjonsalgoritmer er også anvendt innen fysiske roboter. Ved universitetet i Oslo har forskere utviklet sanntids læringsalgoritmer for å trene opp en tobeint pneumatisk kyllingrobot "Henriette". De har anvendt en kombinasjon av en genetisk algoritme og ”hill climbing” algoritme. Målet er å øke robotens evne til å overleve robuste og ukjente forhold[7]. På NTNU anvender Nicolas Bredech et al.[2] evolusjonsalgoritmer for å trene opp små robotagenter som arbeider i en sverm. Agentenes interaksjon med hverandre og omgivelsene er begrenset til enkle regler, men sammen fører svermen til et globalt mønster som evalueres mot ytelsesfunksjoner.

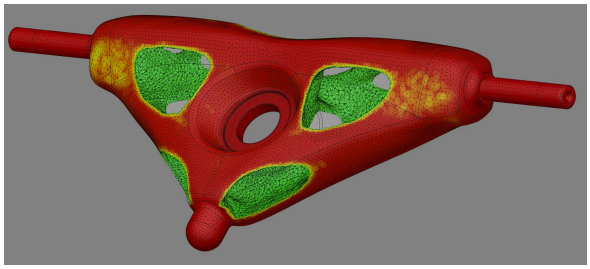
Mekaniske konstruksjoner

Evolusjonsalgoritmer er også anvendt innen optimalisering av mekaniske konstruksjoner. I 2005 sendte NASA opp en mikrosatelitt "Space Technology 5". På satellitten var det montert en kommunikasjonsantenne utviklet med evolusjonsalgoritmer, vist i figur 1.5. Dette er det første objektet utviklet med denne typen kunstig intelligens til å fly i verdensrommet.

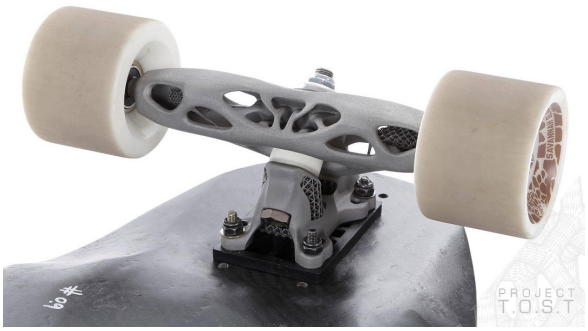


Figur 1.5: ©[Antennas and Propagation Society InternationalSymposium 2005] IEEE[13]

"Additiv tilvirkning" er en konstruksjonsmetode som har hatt stor utvikling de siste årene. 3D-printing i plastikk og metall gjør det mulig å konstruere materialer i komplekse mønster som konvensjonelle maskineringsprosesser ikke klarer. Dette åpner for mer kompliserte mekaniske konstruksjoner som kan optimaliseres med evolusjonsalgoritmer. Konstruksjoner utviklet med evolusjonsalgoritmer er et fagområde kalt "generativt design". Figur 1.6 viser deler til et longboard som er 3D-printet i titan, hvor designet er optimalisert med generativt design av Philipp Manger ved EAH University.



(a)



(b)

Figur 1.6: Prisvinnende titankonstruksjon designet med generativt design.
a) viser 3d modell av konstruksjonen, mens b) viser ferdig produsert komponent
Gjengitt med tillatelse fra Philipp Manger

1.4 Oppgavens målsetning

Målet med oppgaven er å utforske hvordan genetiske algoritmer kan anvendedes for å optimalisere gangmønsteret til en robot. I tillegg er det ønsket å utforske hvordan genetiske algoritmer kan øke en robots evne til å tilpasse seg ukjente omgivelser. Det er dannet en hypotese rundt problemstillingen:

"Kan genetiske algoritmer anvendes for opptrening av sentrale mønstergeneratorer i en firebeint robot?"

I oppstarten vurderte jeg først å simulere et robotsystem for så å implementere bevegelsesparameterne i en praktisk robot. Dette gikk jeg imidlertid bort ifra, og valgte i stedet å fokusere på å kun utføre treningsalgoritmene på en fysisk robot. Dette av to grunner. For det første er oppgaven tidsbegrenset. For det andre kommer den berømte forskjellen mellom teori og praksis. Firebeint bevegelse er en kompleks dynamisk og ulineær problemstilling, hvor tilnærminger og modellfeil vil akkumulere seg opp og totalt bidra til en signifikant forskjell mellom simulert og faktisk system. Alexander Spröwitz et al. [22] har simulert et lignende robotsystem hvor parametere finnes ved Partikkel-sverm optimalisering. De har oppdaget signifikante forskjeller mellom teori og praksis. Spesielt innen modell av friksjonskrefter, ledd og fjær-demper system, samt treghet i servomotorer med høy gir-rate.

Det er altså laget en ekte robot, som må trenes opp til å fungere påvirket av fysikk i den virkelige verden. Siden hypotesen skal utprøves praktisk har dette stilt krav til å finne en metode for lokomosjon som ikke er for kompleks, hvor mange unike bevegelsesmønster kan testes med færrest mulige parametere. Det er funnet at Hopf-stabile oscillatorer kan anvendes for å etterligne CPG som oppfyller dette kravet. Dermed kan robotens gangmønster justeres ved få, men unike parametere. Metoden krever hverken tilbakemelding fra sensorer eller iterativ kalkulasjon av invers kinematikk for baneplanlegging. Oscillatorene er integrert i et nevralnettverk, hvor noder i nevralnettverket trenes opp av genetiske algoritmer.

Andre har også arbeidet med optimalisering av CPG nettverk med evolusjonsalgoritmer i simulerte miljøer, som Lewis et al.[12] og Miguel Oliveira et al.[17]. Denne oppgaven har derimot en praktisk tilnærming til problemet.

Oppgaveteksten er:

"Optimalisering av gangmønster for firebeint robot med en genetisk algoritme".

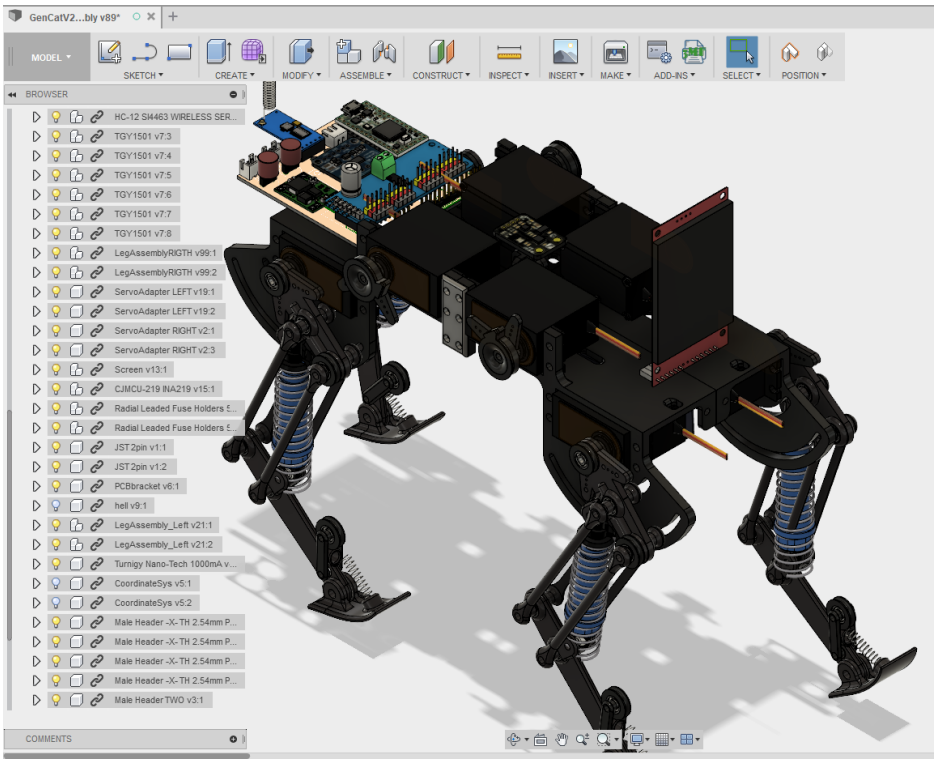
Kapittel 2

Teori

2.1 Verktøy for robotkonstruksjon

3D-tegneverktøy, CAD

Robotens hovedkonsruksjoner er tegnet i 3D-tegneverktøyet Autodesk Fusion 360.[1] Robotens komponenter tegnes hver for seg, og alle delene legges inn i et felles "assembly", vist i figur 2.1. Komponentene settes sammen ved å definere delenes relative posisjoner. Lineære og roterende koblinger defineres slik at bevegelige sammenhenger kan simuleres.



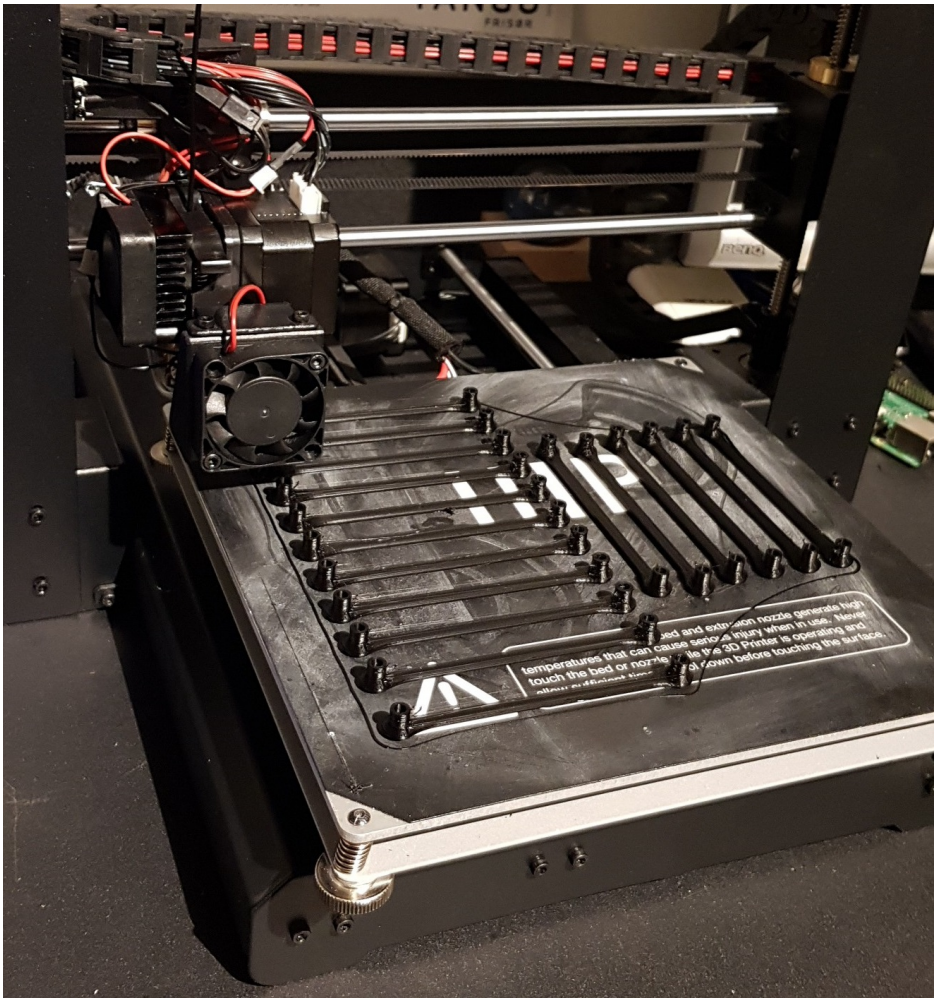
Figur 2.1: Skjermdump av robotsammenstilling i 3D-tegneverktøyet Autodesk Fusion 360

[1] https://www.autodesk.com/products/fusion-360

3D-printede deler

Robotens hovedkomponenter er produsert med en 3D-printer av type Monoprice Maker Select V2. 3D-printing er en additiv konstruksjonsteknikk som muliggjør rask utvikling gjennom generasjoner av prototyper.

Bilde 2.2 viser deler til robotbeinene under konstruksjon i 3D-printeren.



Figur 2.2: 3D-printer under konstruksjon av deler til robotens bein

For å 3D-printe robotens deler blir CAD modellene konvertert fra 3D til lagsis 2D-modeller via "slicer"-programmet Cura.[2] 3D-printeren varmer så plastikk opp til smeltetemperatur med en ekstruder hvor ekstruderen tilfører plastikken lagsvis fra bunn til topp.

Kretskortdesign

Kretskort som er utviklet er tegnet i designerverktøyet Autodesk Eagle v.8.7.0.[3] Programmet har bibliotek for alle standardkomponenter, mens noen komponenter må tegnes inn manuelt. Først tegnes et koblingsskjema hvor komponentenes spenningstilførsler og sammenkoblinger defineres. Deretter legges komponentene ut på en 2D tegning av kretskortet hvor kretskortbanene og gjennomføringer legges til. Det er en kobling mellom Autodesk Eagle og Autodesk Fusion 360 slik at 3D-modell av kretskort automatisk synkroniseres med 3D-modell av roboten.

Designet er sendt til produksjon i Kina av Elecrow Technology.

[2] `https://ultimaker.com/cura`
[3] `https://www.autodesk.com/products/eagle/free-download`

2.2 Bevegelsesmønster

Dette kapittelet beskriver hvordan Hopf-oscillatorer kan anvendes for en forenklet matematisk modell av en sentral mønstergenerator (CPG). Å anvende Hopf-oscillatorer gir muligheter for å styre lokomosjon hos både to og firebeinede roboter. For eksempel kan flere oscillatorer kobles i serie for å kunne gjenskape mer kompliserte bevegelsesmønsteret. L. Righetti og Auke Jan Ijspeert[18] foreslår hvordan en sum av oscillatorer kan trenes opp til et kjent signal, som kan være signaler målt fra beinbevegelser til dyr. En sum av adaptive oscillatorer kan så trenes opp for å bli tilnærmet lik det ønskede signalet, lignende en Fourier serie representasjon. I denne oppgaven er det imidlertid fokus på å finne en forenklet modell. Det er valgt å representere hvert bein med en oscillator.

2.2.1 Hopf-oscillator

En Hopf-avgrening er definert som det punktet hvor et system beveger seg fra en stabil tilstand og over i en marginalt stabil tilstand hvor en oscillasjon oppstår rundt et lokalt punkt.

System som passerer det sub-kritiske punktet i Hopf-avgreningen går over til å ha rene imaginære komplekskonjugerte poler som gir en stående oscillasjon rundt punktet. De stående periodiske svigningene som oppstår defineres som en ”limit cycle”, og kan nyttes som en signalgenerator. Dette gir en ”Hopf-oscillator”.

Det unike med oscillasjonen er at hvis en påtrykker systemet en forstyrrelse, konvergerer det tilbake til den periodiske løsningen. Oscillatoren som er valgt er hentet fra Simon Rutishauser et al.[19] som har laget et lignende robotsystem med koblede Hopf oscillatorer.

Å anvende Hopf oscillatorer for å representere sentrale mønstergeneratorer oppfyller kjente egenskaper for bevegelse:

- Kjent rotasjonsretning
- Syklisk/periodisk/rytmisk
- Stabilitet ved påtrykte forstyrrelser
- Trenger ikke tilbakekobling av sensor
- 2 faser i bevegelsesmønsteret: Standfase og pendelfase. Hver fase kan ha ulik tidskonstant.
- Hastighet og svingamplitude kan endres under bevegelse, hvor oscillatoren konvergerer mot nye mønster kontinuerlig.
- Fasekobling mellom beinene kan endres i sanntid.

Det er mulig å legge inn offset for styring av balanse, men det er ikke implementert i denne oppgaven. Andre lignende oscillatorer som Matsuoka’s eller Rayleigh’s kan gi ikke-sirkulære mønster, foreslått av Julien Nicolas[16].

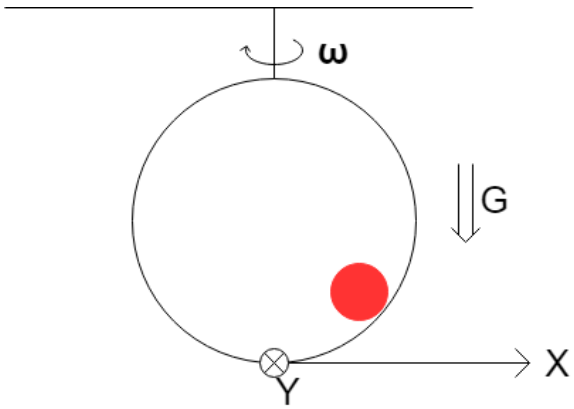
Oscillatoren gitt ved ligning 2.1 består av to koblede ulineære differensialligninger. Systemet har stående oscillasjoner i de to dimensjonene x og y.

$$\begin{aligned}\dot{x}_i &= \alpha(\mu - r_i^2)x_i - \omega_i y_i \\ \dot{y}_i &= \beta(\mu - r_i^2)y_i + \omega_i x_i + \sum_j k_{ij} y_j \\ \omega_i &= \frac{\omega_{stand}}{e^{-by}+1} + \frac{\omega_{pendel}}{e^{by}+1}\end{aligned} \tag{2.1}$$

Hvor:

x, y	= Utganger fra oscillator.
$\mu > 0$	= Ønsket amplitude på utgangssignal (kvadrert)
r	= $\sqrt{(x^2+y^2)}$
α, β	= Konstanter for konvergeringstid
ω_i	= Oscillasjonsfrekvens
ω_{pendel}	= Ønsket oscillasjonsfrekvens, pendelfase
ω_{stand}	= Ønsket oscillasjonsfrekvens, standfase
b	= Konstant for overgang mellom faser
$\sum_i k_{ij} y_j$	= Skyver systemet til ønsket faseforskyvning
k_{ij}	= Koblingsmatrise

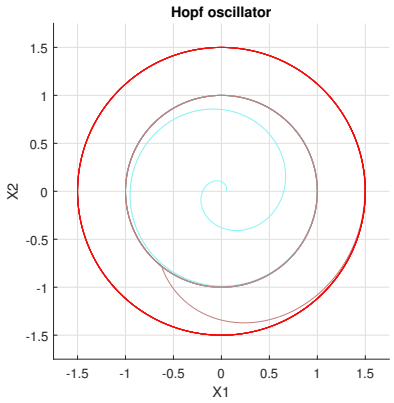
Figur 2.3 viser et eksempel på et system med stående oscillasjoner som kan defineres som en Hopf-oscillator. Vi kan se for oss en ball i en roterende rigid og hul sfære. Sfæren starter å rotere med en konstant hastighet ω, hvor ballens posisjon i to dimensjoner måles. Ballen vil slynges oppover ytterveggen, hvor svingningsamplituden i X og Y retning vil gå mot en endelig verdi. Numerisk må en "så" en initialverdi x eller y $\neq 0$ for å starte oscillasjonene. Nå er ikke dette systemet identisk med systemet i ligning 2.1, men de baserer på samme stabilitetsprinsipper. (Eksempelet er hentet fra J. E. Marsden og M. McCracken "The Hopf Bifurcation and Its Applications", 1976[15])



Figur 2.3: Eksempel på et oscillerende system.
Kule i roterende sfære gir stående svingninger i to dimensjoner
Vinkelhastighet ω gir radien til oscillasjonene i X og Y dimensjon.

Oscillatoren er implementert i MATLAB for å visualisere hvordan signalet utvikler seg. Figur 2.4 viser utviklingen.

Initielt er μ lik 1, men endres til $1,5^2$ etter t = 6 sekunder. Utgangen endres kontinuerlig mot den nye radien. Øvrige parametere: α og $\beta = 2$, b = 100, $\omega_{pendel} = 3$, $\omega_{stand} = 9$,

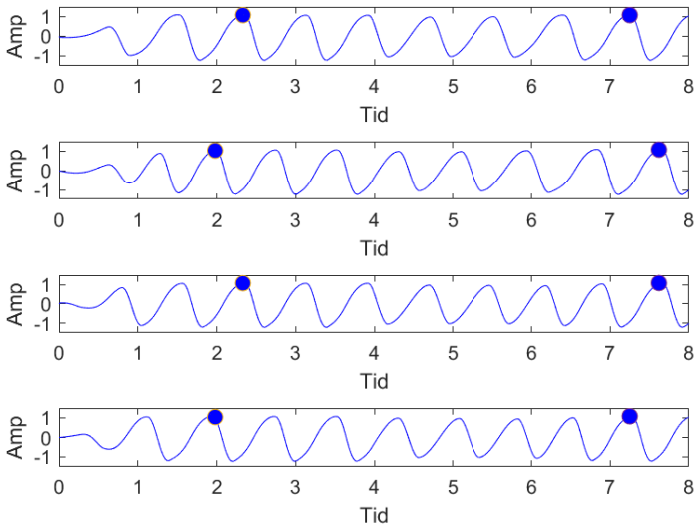


Figur 2.4: Hopf oscillatorens utvikling ved endring av amplitude

Koblingsmatrisen K sørger for at robotens fasekobling mellom beinene (ganglag) opprettholdes. Ganglaget er gitt av hvilken koblingsmatrise som anvendes. Koblingsmatrisen kan endres i sanntid.

$$K = \underbrace{\begin{pmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{pmatrix}}_{\text{Trav (Trot)}} \quad \underbrace{\begin{pmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{pmatrix}}_{\text{Passgang (Pace)}} \quad \underbrace{\begin{pmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{pmatrix}}_{\text{Bundet (Bound)}} \quad \underbrace{\begin{pmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{pmatrix}}_{\text{Skritt (Walk)}}$$

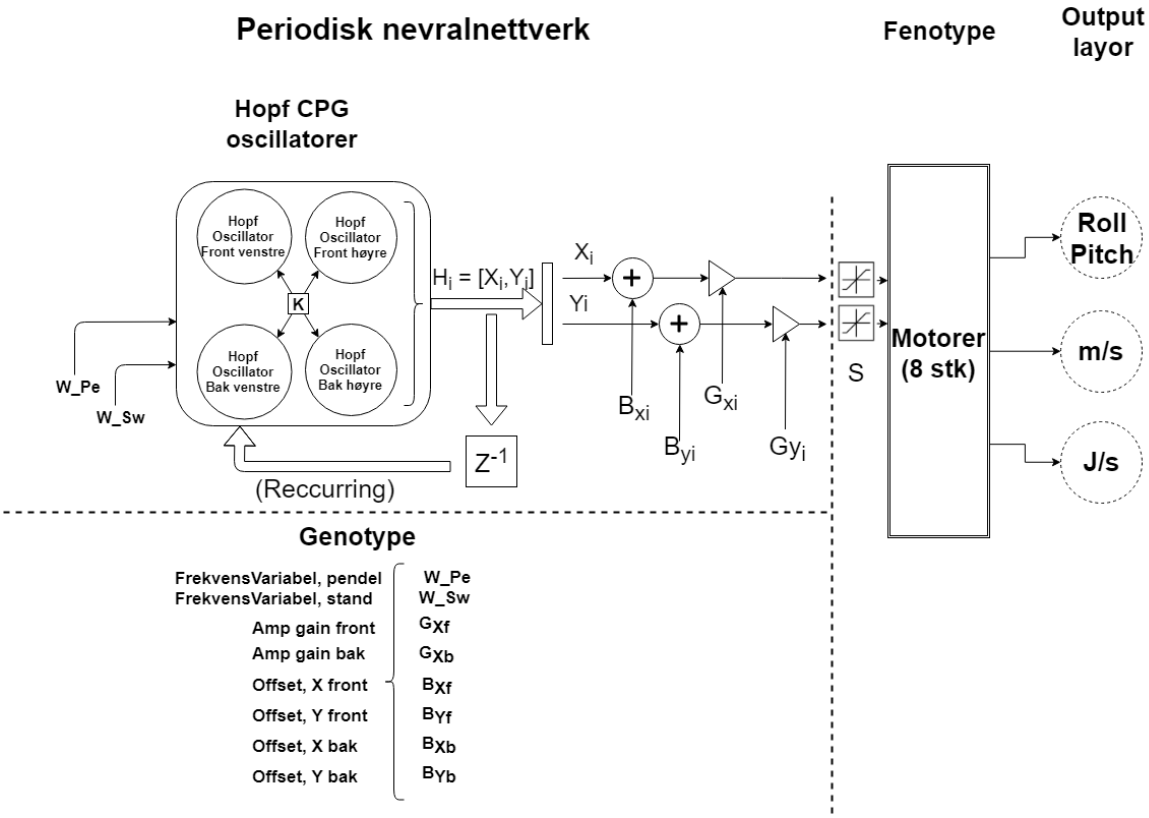
Figur 2.5 viser hvordan ganglaget kan skiftes i sanntid. Roboten starter gange med koblingsmatrise for passgang. Hver oscillator justerer seg selv mot riktig faseforskjell, etter t = 2 sekunder. Ved t = 4 sekunder endres koblingsmatrisen fra passgang til trav, og ved t = 7 sekunder har roboten justert inn faseforskyvningen korrekt tilpasset det nye ganglaget.



Figur 2.5: Fasekobling under endring av gangmønster fra passgang til trav.
Her plottes X verdien fra oscillatorene.
Koblingsmatrise K endret ved t = 4

2.3 Periodisk nevralnettverk

De koblede Hopf-oscillatorene implementeres i et kunstig nevralnettverk. Nevralnettverket er av type periodisk ("recurret") pga. tilbakekobling i Hopf-oscillatorene. Parametere til Hopf-oscillatorene, samt forsterkning og skyv (offset) til motorene defineres som innganger eller noder i nevralnettverket. Det er disse inngangene og nodene som trenes opp med genetiske algoritmene. Dette er beskrevet i neste delkapittel. Enhver unik genkombinasjon i nevralnettverket resulterer i unike sykliske bevegelsespådrag til motorene (fenotype). Dette igjen fører til målbare ytelser (hastighet, stabilitet og energiforbruk). De målbare ytelsene gir nevralnettverkets utgangsnoder.



Figur 2.6: Nettverkstruktur i periodisk nevralnettverk. Parameter i genotype gir et syklisk motorpådrag (fenotype), som igjen fører til målbare ytelser.

Forklaring til figur

Genotype:

W_{sw}, W_{pe} = Frekvensvariabler som styrer tid for pendelfase og svingfase
G_{Xf} = Forsterkning, hofte bak
G_{Xb} = Forsterkning, hofte foran
B_{Xf} = Forskyvning, hofte bak
B_{Yf} = Forskyvning, kne foran
B_{Xb} = Forskyvning, hofte bak
B_{Yb} = Forskyvning, kne foran

Øvrig:

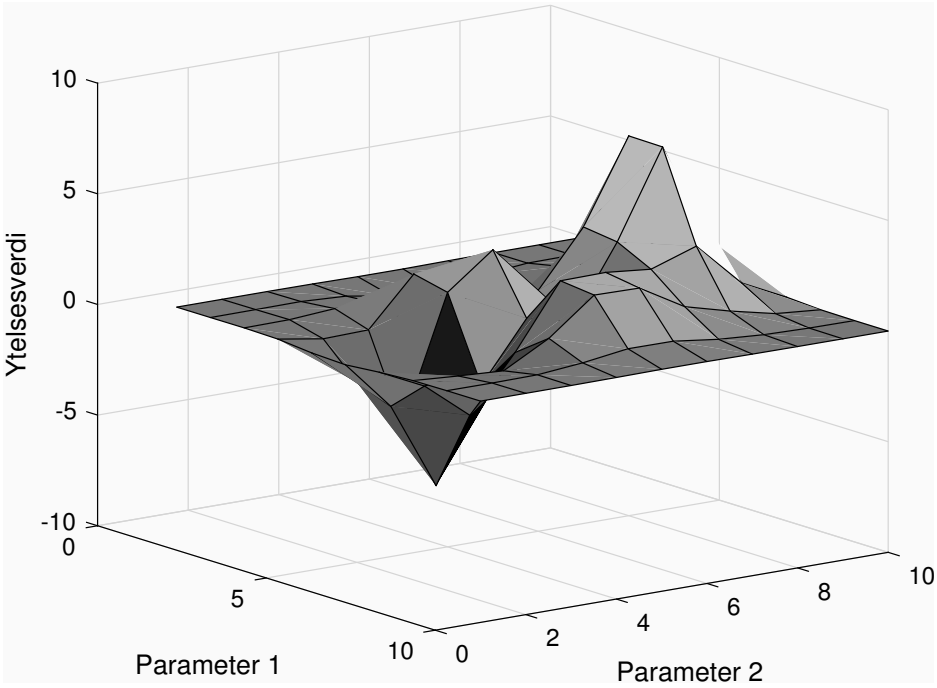
H_i = Utgangen til oscillator i (i representerer bein 1 - 4)
X_i = Hofteposisjon, bein i
Y_i = Kneposisjon, bein i
K = Kobling mellom oscillatorene sørger for å opprettholde faseforskjell mellom bein
S = Metning for motorenes maksimalposisjoner
Z^{-1} = Tilbakekobling fra forrige tidssteg

2.4 Treningsmetode: Genetiske algoritmer

Genetiske algoritmer er en stokastisk optimaliseringsmetode først introdusert av Holland [8], 1975. Metoden har utspring fra Darwins evolusjonsteori, hvor hensikten er å søke etter globale optimale løsninger i et komplisert multidimensionelt løsningrom. Genetiske algoritmer er en avgrening under "reinforcement" maskinlæring, igjen avgrenet under kunstig intelligens (AI).

Genetiske algoritmer er typisk anvendt der det er for komplekst å prøve ut alle mulige kombinasjoner av parameter i en funksjon, hvor det trolig er flere lokale maksima. Ytelsesfunksjonen trenger ikke å være kontinuerlig. Genetiske algoritmer har en unik egenskap for å raskt finne globale toppunkt i et gitt søkeområde. Funksjoner som mutasjon og parameterkrysning gjør at algoritmen har en eksperimentell og uforutsigbar fremgangsmåte, hvor andre optimaliseringsmetoder (som gradientsøk) kan komme til kort[23]. I figur 2.7 er det visualisert en ytelsesfunksjon gitt to funksjonsparameter. Gradientsøk kan her konvergere mot et lokalt topp-punkt, uvitende om andre toppunkt finnes i løsningsrommet.

I denne oppgaven anvendes genetiske algoritmer for å utvikle nodene (genotype) i nevralnettverket figur 2.6 mot verdier som gir bedre ytelse målt mhp. lokomosjon.



Figur 2.7: Figuren viser en løsningsrommet gitt to diskrete variabler. Genetiske algoritmer kan hoppe mellom topper i løsningsrommet. Diskretisering gir en grovere oppdeling av ytelsesfunksjonen.

2.4.1 Beskrivelse av treningsalgoritmen

Populasjon

En populasjon består av de ulike individer som lever i samspill med hverandre i en generasjon. Individene er utviklede avkom av forrige generasjon, og antallet i hver populasjon er holdt konstant. Individene i populasjonen har hver seg unike egenskaper (fenotyper), gitt dens sammensetning av gener i kromosomet. Det er individene (kromosonene) i populasjonen som blir utviklet igjennom generasjoner.

Antall individer i hver populasjon beskriver hvor mange parametersett som testes parallelt, og må tilpasses slik at opptreningstiden kan utføres innen rimelighetens grenser.

Kromosonene tilhørende en populasjon er representert med en vektor med integerverider. Den initielle populasjonen P(0) gis tilfeldig valgte parametere i deres variasjonsområde.

Genotype og fenotype

Hvert individ i populasjonen har som nevnt unike egenskaper som er gitt av genkombinasjonen i kromosomet. I algoritmen er kromosomet representert som en vektor med data for parameterne som beskriver robotens bevegelsesparameter.

Genotype er altså den unike kombinasjonen og rekkefølgen av data som er lagt i kromosom-vektoren, mens fenotype representerer mer overordnet konsekvensene genkombinasjonen har i den fysiske verden. I den naturlige evolusjonsprosessen kan eksempler på fenotyper være hudfarge, øyenfarge etc, mens genotypen beskriver rekkefølgen av de fire nitrogenbasene i DNA-stigen, A (adenin), C (cytosin), T (thymin) og G (guanin).

I denne oppgaven gir genotypen noder i nevralnettverket som påvirker bevegelsesmønsteret til beinene. Fenotypen kan beskrives som det todimensjonale sykliske bevegelsesmønsteret, gitt av genotypen.

Variasjon

Variasjonen til data i et gen beskriver hvilket område parameterne har mulighet for å strekke seg i, og også eventuelt hvilken oppløsning variablene har. Variasjonen til genet har stor innvirkning på antall mulige løsninger i løsningsrommet, og påvirker dermed også opptreningstiden betraktelig.

Det er viktig å ha tilstrekkelig oppløsning i parameterne. Dermed unngår vi at optimale løsningsparametere ligger imellom stegene i oppløsningen og dermed aldri nåes. I figur 2.7 kan man se hvordan oppløsningen er med på å diskretisere ytelsesfunksjonen.

Variasjonsområde avgrenses til det område hvor det er forventet at optimale løsninger vil finnes.

I vårt tilfelle er det valgt å ha en oppløsning på 100 i hver av de 8 genene, som skal være nok for å ikke diskretisere vekk toppunkt i ytelsesfunksjonen. Dermed finnes det teoretisk 100^8, 10^{16}, antall løsninger. Mange av disse løsningenene er tilnærmet identiske, og det er mulig en grovere diskretisering hadde vært tilstrekkelig. Finere oppløsning gir derimot ikke direkte lengre opptreningstid.

Ytelsesmål

Ytelsesfunksjonen $f(K)$ beskriver objektivt robotens ytelse, gitt funksjonens inngangsparametere (altså kromosomet med bevegelsesparameterne). Den "perfekte løsningen" (globalt optima) er ikke kjent. Ytelsesmålet defineres da på en måte hvor de kan sammenlignes med de andre parameternes ytelser i opptreningsfasen. I den gjeldende problemstillingen er det ikke mulig å beregne ytelsene, de må måles via eksperimentell test. Det er valgt å eksperimentere med å tilegne ytelse til egenskaper som beskriver god lokomosjon.

- Gjennomsnittlig hastighet, $\overline{v}$
- Stabilitet, $Var(\phi) + Var(\theta)$
- Energiforbruk, $\overline{w}$

Under testing i kapittel 4 er det gjort eksperiment med de forskjellige ytelsesmålene. De forskjellige ytelsene vil gi forskjellig opptrent gange til roboten, og valg kan tas i forhold til subjektivt ønsket robotbevegelse.

Seleksjon

For at en populasjon skal utvikle seg gjennom generasjoner må det gjøres en seleksjonsprosess. I naturen er det kun de artene som er best tilpasset omgivelsene som overlever, formerer seg, og fører sine gen videre til neste generasjon. For algoritmen i denne oppgaven er det valgt å benytte en metode kalt "*Rank space method*" for selektering.

Etter evaluering av en populasjon blir alle kromosom fordelt i et såkalt "mating pool". Antall representasjoner av hvert kromosom er gitt av den relative ytelsen i forhold til de andre kromosomene i populasjonen. Dette gir en lineær sannsynlighetsfordeling. Individer med høyest ytelse er mest representert, og har dermed størst sannsynlighet for å være med videre til neste generasjon.

Sansynligheten for at et individ skal videreføres i krysningsprosessen er gitt av:

$$P(n) = \frac{(K+1) - rank(n)}{\sum_k^K rank(k)} \tag{2.2}$$

Hvor:

$P(n)$ = Sansynlighet for å trekke kromosom n som foreldre
$rank(n)$ = Kromosomets rangering av ytelse, rank = 1 representerer beste ytelse
K = Populasjonsstørrelse

Eksempel:

Populasjonsstørrelse = 10

Sansynligheten for å velge det 2. beste kromosomet som foreldre:

$$P(2) = \frac{11-2}{1+2+3+4+5+6+7+8+9+10} = 0,1636$$

Krysning

Etter individene i populasjonen har blitt målt og lagt i et ”mating pool”, lages neste generasjon ved hjelp av en krysningsalgoritme. Det blir plukket to individer fra ”mating pool”-en, hvor sannsynligheten for å velge et kromosom er gitt av ligning 2.2. Det er valgt å ikke implementere tilbakelegging, det vil si at et avkom ikke kan ha mor og far fra samme kromosom. Dette fører til at utviklingen ikke konverger for raskt mot dominerende parametervalg.

Parameterne som det arves ned til velges ved å utføre en ”cross over” operasjon på foreldreparet. Det velges et tilfeldig punkt i genet, hvor den ene siden kopierer gener fra det en forelder, mens den andre siden av punktet får kopiert gener fra den andre. Hvilken side av genet som kommer fra hvilken foreldre er tilfeldig valgt.

Krysningsalgoritmen utføres til den nye populasjonen P(k+1) har like mange individer som foregående populasjonen P(k).

Mutasjon

Mutasjon er viktig for å introdusere utvalg av variasjonen som ikke er til stede i arvmaterialet til noen av foreldrene. I naturen kommer mutasjon inn ved en feil i gen-kopieringsprosessen, eller av en ytre påvirkning som f.eks radioaktiv stråling. Denne egenskapen er veldig viktig siden det introduserer mulighet for at avkommet av foreldreparret får en gen som foreldre ikke har. Dette gir genetiske algoritmer en unik evne til å ”hoppe” i løsningsrommet. Praktisk i algoritmen utføres mutasjonsprosess ved å tilfeldig bytte ut et gen i et kromosom innen for variasjonsområde, etter den nye populasjonen er valgt.

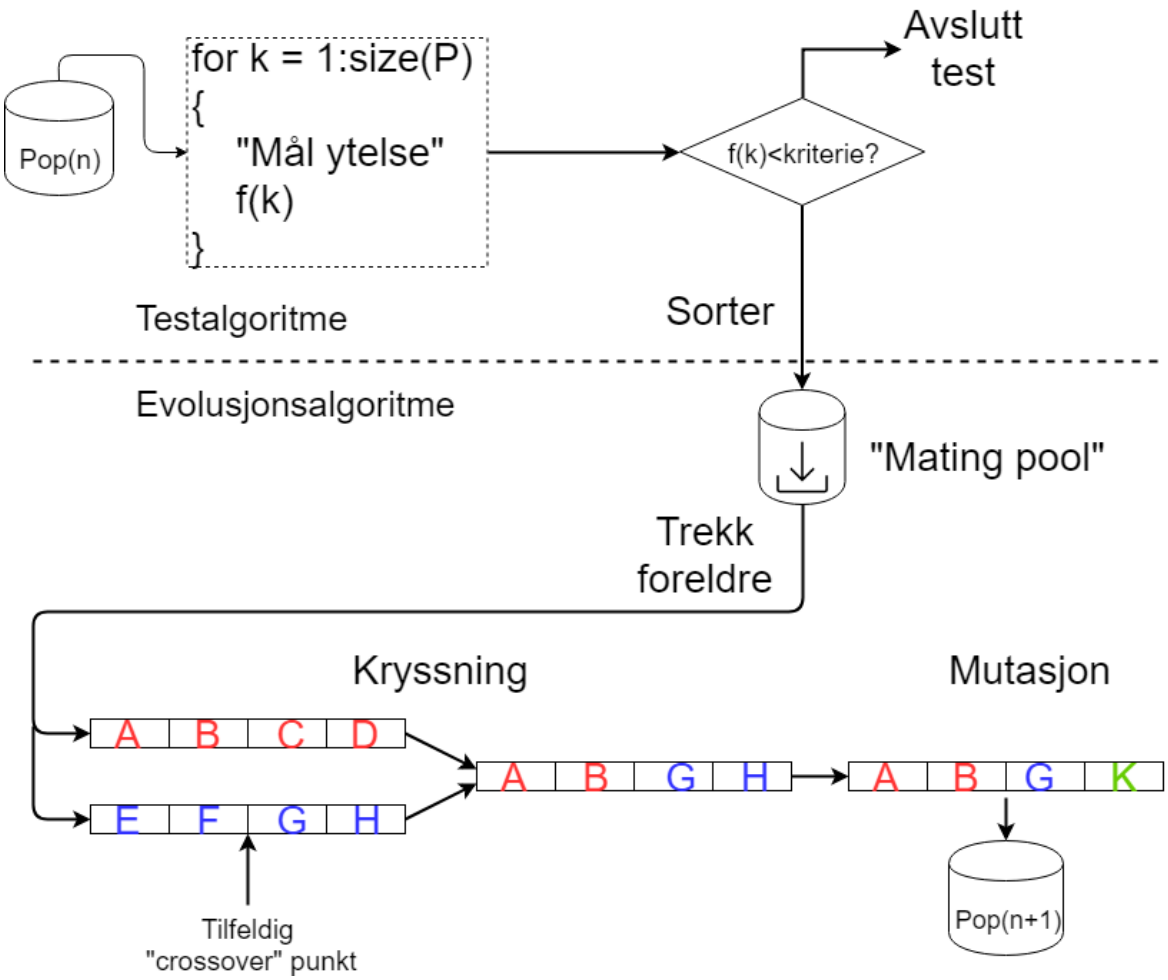
Mutasjonsgrad må velges med omhu. Hvis den velges for høyt vil det bli introdusert hvor mye støy i parameterne under utviklingen. I de tilfellene hvor det er for liten grad av mutasjon vil algoritmen konvergere mot de initielle parameterene i den første populasjonen. Erfaring tilsier at mutasjonsgraden bør ligge mellom 2 og 5 %. I dette tilfellet er populasjonstørrelsen begrenset grunnet praktisk trening. Da bør mutasjonsraten være høy for å introdusere parametere som ikke var tilstede i den initielle populasjonen.

Figur 2.8 viser hvordan kromosomene i en populasjon utvikler seg igjennom generasjoner. Treningnen avsluttes når et ytelseskriterie er nådd.

Eksempler på ytelseskriterier

- En ønsket ytelse er oppnådd
- Et forhåndsbestemt antall generasjoner har blitt utviklet
- Generasjonens beste ytelse har ikke økt de siste n generasjonene

Den parameterkombinasjonen som har fått best ytelse igjennom hele treningen. Det er ikke nødvendigvis parametere fra siste generasjon som har fått den beste ytelsen, avhengig av valgt metode for å stanse trening.



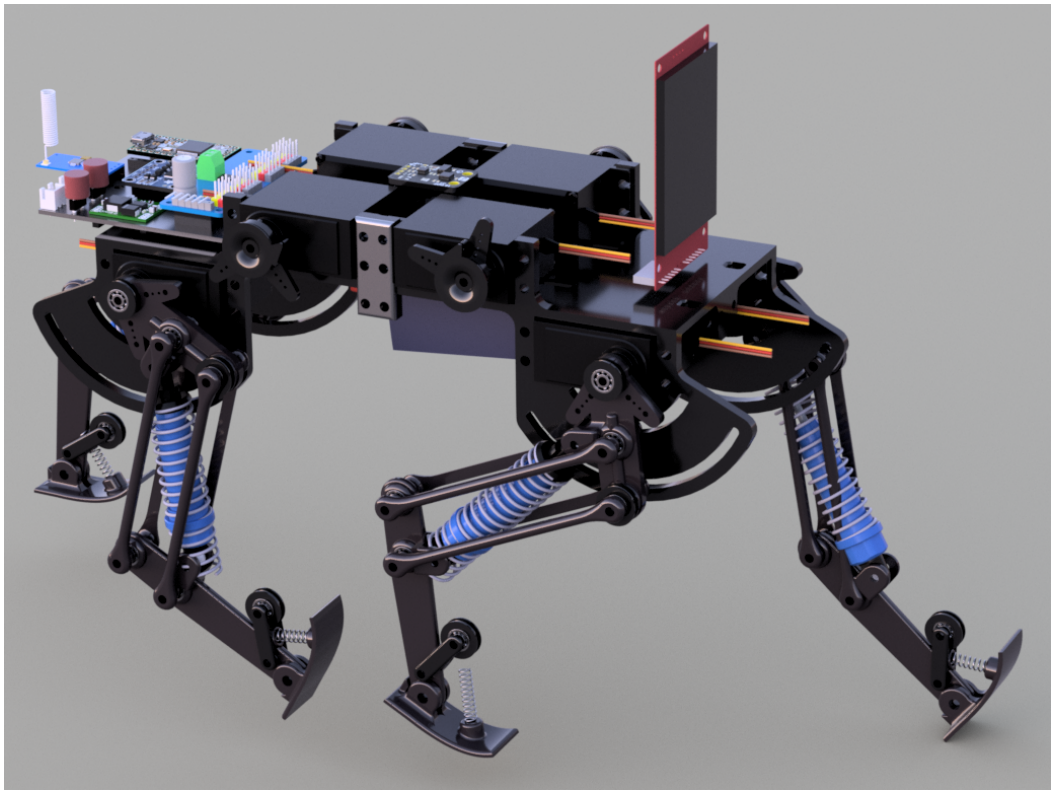
Figur 2.8: Flytdiagram over genetisk algoritme
Diagrammet viser kromosomenes utvikling igjennom generasjoner n
Genene er her representert som bokstaver, i roboten representeres de som heltall

Kapittel 3

Roboten

Det primære målet med oppgaven er å utvikle og teste genetiske algoritmer for optimalisering av gangen til en mobil firbeint robot. For å gjøre dette trengs en fysisk robot. Dette kapittelet beskriver hvordan en slik robot er designet og bygget.

Roboten skal primært brukes for å eksperimentelt trene opp sine bevegelsesparametere. Derfor har det vært fokus på å konstruere en robotplattform som er stabil og robust, hvor roboten ikke velter selv om bevegelsesparameterne ikke er optimale. (Figur 3.1).



Figur 3.1: CAD modell av mobil firebeint robotplattform

Robotens hoveddesign er inspirert av arbeid utført ved Biorobotics Laboratory ved universitetet EPFL i Sveits[1].

Det presiseres at konstruksjonen til roboten er inspirert av designet til robotene Cheetah-cub og Oncilla ved EPFL, men alle CAD-tegninger, systemvalg innen elektronikk og mekanikk, dimensjonering, programkode og opptreningssystem er eget arbeid gjort i forbindelse med denne oppgaven.

[1] `https://biorob.epfl.ch/`

3.1 Mekanisk konstruksjon

Roboten er konstruert som en mobil firbeint "Quadruped"-robot. Delkapittel 2.1 beskriver hvilke verktøy som er brukt under utviklingen av roboten.

Lokomosjon av robotens fire bein styres av til sammen åtte motorer. Hvert bein består av to motorer som styrer hofte og pantografsystem. Den ene motoren styrer hoftens vinkelposisjon direkte koblet på motor i hoftens sentrum. Den andre motoren sitter på robotens overkropp, og trekker til seg en vaier for posisjonstyring av et pantografsystem.

Pantografsystem fungerer ved at servomotor tiltrekker seg pantografen med vaier (3.2) hvor et fjær-demper system holder motkraft til systemet. Fjær-dempersystemet reduserer også støt under gange.

Hoftene og pantograf-system kan kjøre tilnærmet uavhengig av hverandre til en posisjon gitt av en sentral styreenhet.

Robotens vektbudsjett er oppsummert i tabell 3.1.

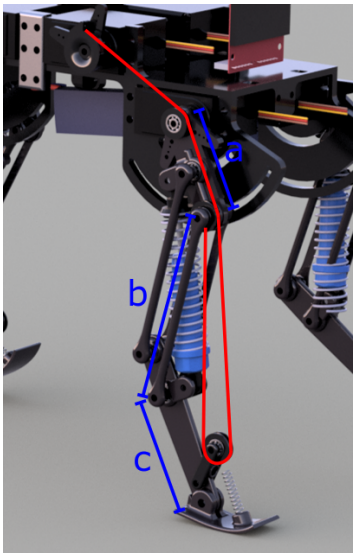
Frambeinene og bakbeinene er identiske.

Tabell 3.1: Vektbudsjett

Kropp	103 gram
Beinkonstruksjon	198 gram
Motorer	504 gram
Elektronikk	180 gram
Batteri	51 gram
Total	**1036 gram**

Tabell 3.2: Dimensjoner

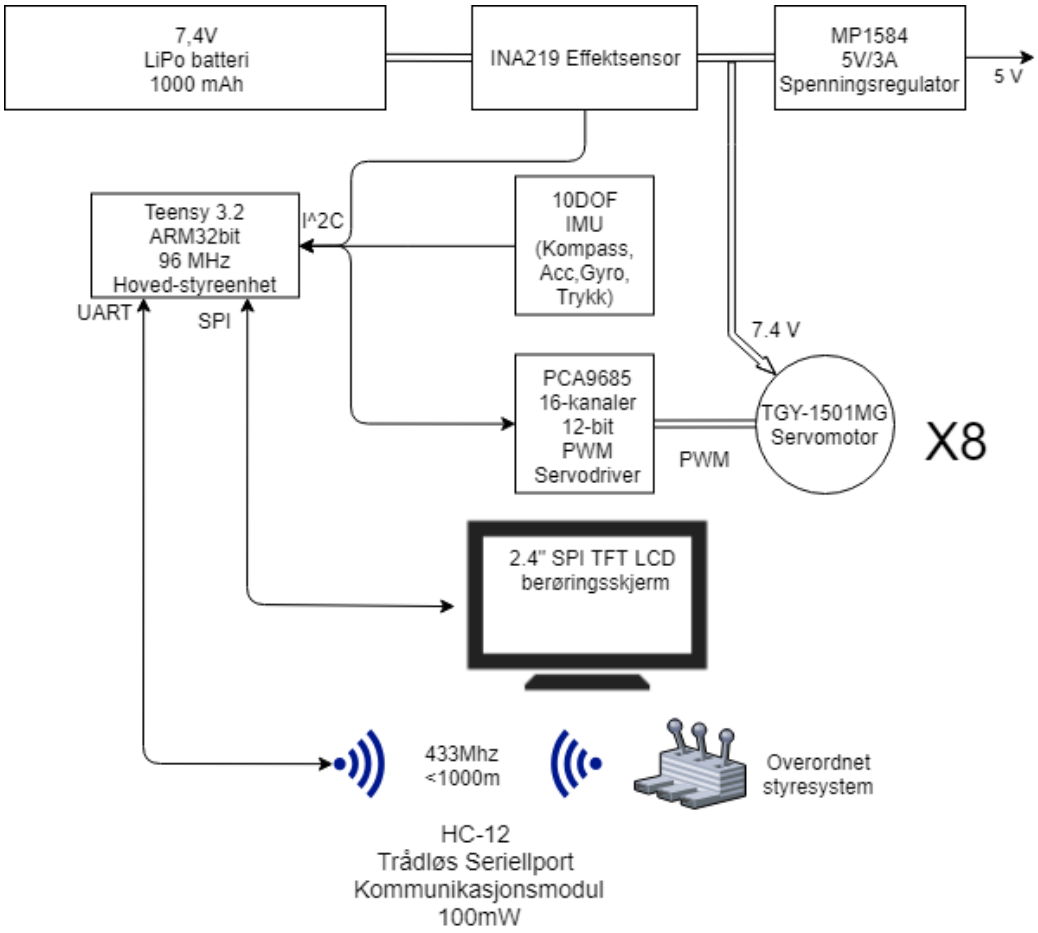
3 segmented pantograf	
Hofte (a)	42 mm
Parallelstag (b)	85 mm
Legg (c)	50 mm
Øvrige mål	
Bredde	117 mm
Lengde	255 mm
Høyde (kalibreringsposisjon)	203 mm



Figur 3.2: 3D CAD modell av beinkonstruksjon, rød vaier er fremhevet

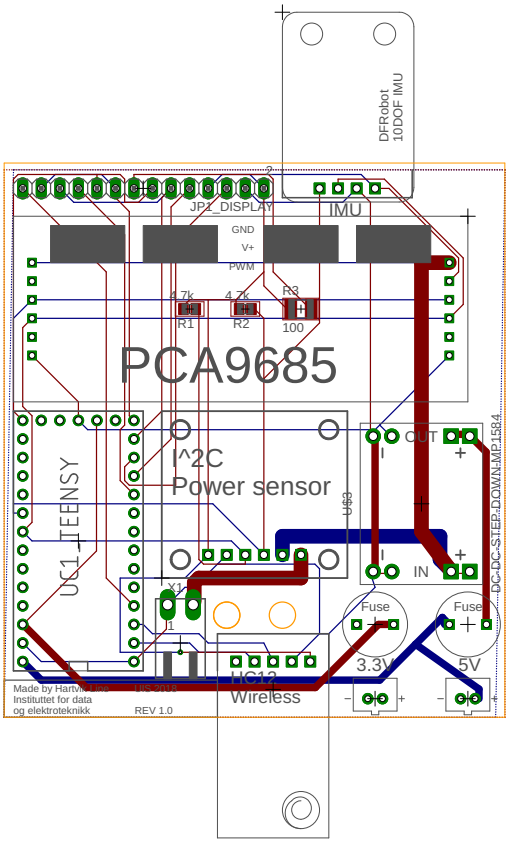
3.2 Hardware

Figur 3.3 viser et enlinjeskjema over robotens komponenter som er valgt og grensesnitt. Datablad tilhørende komponentene ligger vedlagt som komprimert fil, se side 50.

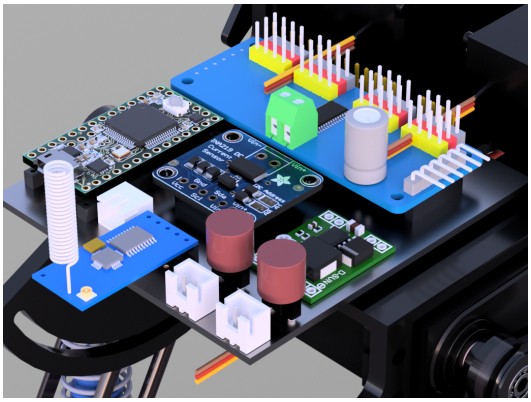


Figur 3.3: Enlinjeskjema over robotens grensesnitt

Alle komponentene er koblet sammen i et felles grensesnitt-kretskort. Figur 3.4 viser 2D CAD modell av utlegget til kretskortet fra programmet Autodesk Eagle. Figur 3.5 viser grafisk fremstilling av 3D CAD modell i programmet Autodesk Fusion 360.



Figur 3.4: CAD tegning av kretskortutlegg



Figur 3.5: 3D modell av kretskort med alle komponenter lagt til i 3D modell av roboten

Mikrokontroller

Robotens mikrokontroller Teensy 3.2 støtter programkode for Arduino i C og C++. Dermed finnes det mange biblioteker for grensesnitt med andre komponenter ferdig utviklet i åpen-kildekode.

Tabell 3.3: Utviklingskort for mikrokontroller

Utviklingskort	Teensy 3.2
Mikrokontroller	32 bit ARM Cortex-M4
Klokkehastighet	96 MHz
BUS	CAN, SPI, I^2C, USB
Spenningsnivå I/O	3.3 V (5V tolerant)
Flashminne	256 kB
Ram-minne	64 kB
EEPROM-minne	2 kB
Vekt	2.7 gram

Motorer

Til servomotorer er det valgt relativt rimelige girede servomotorer: TURNIGY™ TGY-1501MG MG SERVO 25T. Motorene er typsik brukt til fjernstyrte hobbyprosjekter. Vinkelsetpunkt styres mellom 0 og 180 grader ved PWM signaler[2]. Motordata er beskrevet i tabell 3.4.

PWM signalene genereres av enhet PCA9685 PWM driver. Setpunktene blir sendt over I^2C buss mellom mikrokontrolleren og PWM driveren.

Tabell 3.4: Motordata

Motorer	TURNIGY™ 1501MG
Oppgitt moment	17.0 $Kg * cm$, $\simeq$ 1.66 Nm
Oppgitt hastighet	0.14 $\frac{s}{60°}$
Vekt	63 gram

Trådløs kommunikasjon

Roboten kommuniserer med overordnet kontrollenhet via en trådløs sender og mottaker type HC-12. Modulen sender serielldata på 433.4 MHz båndbredde med en bitrate på 19200 bps.

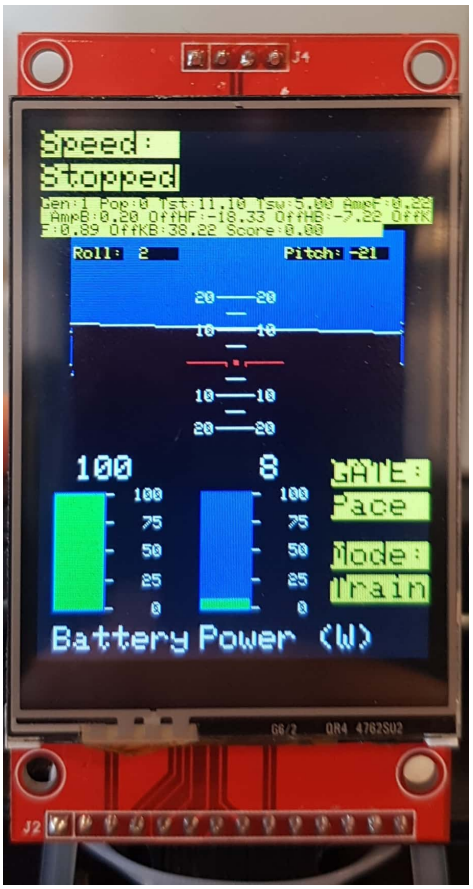
Batteri

Tabell 3.5: Batteri

Batteri	Turnigy nano-tech
Nominell spenning	7.2 V
Celler	2
Type	Lithium polymer
Kapasitet	1000 mAh
Utlading	20 amp kont
Dimmesjoner	101x20x11mm
Indre motstand	1.2$m\Omega$
Totalvekt	51 gram

2,4 tommers LCD berøringsskjerm

Fremvisning av sensordata, treningsopplysning, samt grensesnitt via berøringsskjerm.



Figur 3.6: Berøringsskjerm viser informasjon av blant annet treningsstatus, akselerometerdata, batterikapasitet og effektforbruk

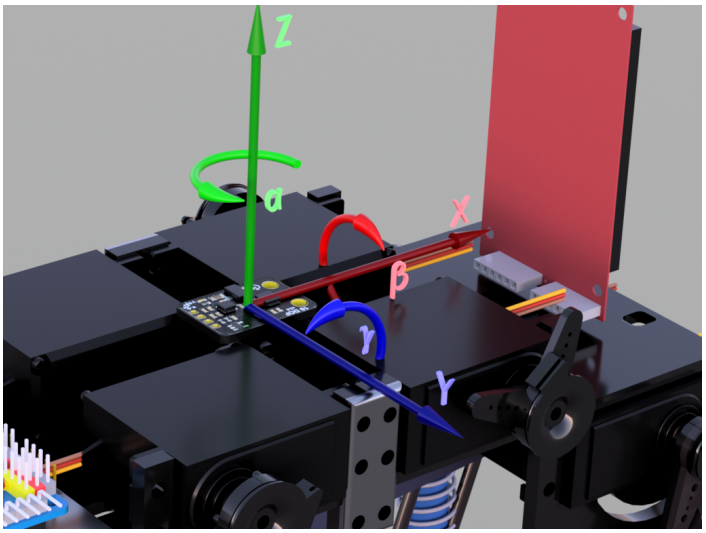
[2]PWM-protokoll: Data er gitt av hvor lenge signalet har høy flanke i løpet av en periode på 20 ms. Motorene styres lineært mellom 1000 og 2000 µs per periode, og reguleres til setpunkt med intern med PD regulator.

Sensorer

Sensorene er valgt så roboten selv kan bedømme egenskaper for lokomosjon: Hastighet, balanse og energiforbruk. (Hastighet beregnes med ekstern sensor, se delkapittel 4.1.)

IMU

Det er montert en treghetssensor (IMU) i robotens massesenter. Under opptrening er roll og pitch data fra treghetssensoren anvendt for å beregne ytelsesmål innen robotens stabilitet. IMU sensoren består av et utviklerkort som integrerer tre sensorer: Gyro (ITG-3205), akselerometer (ADXL345) og kompass (HMC5883L). Totalt måles akselerasjonskrefter i 6 frihetsgrader, elektronisk kompassretning i tre frihetsgrader og høyde (meter over havet). Sensordata vises frem i en kunstig horisont i berøringsskjermen, vist i figur 3.6. Grensesnittet mellom utviklerkort og mikrokontrolleren er I^2C bus.



Figur 3.7: Akselerasjoner måles i translatorisk retning i X Y og Z. med akselerometer. Vinkelakselerasjoner måles med gyro, hvor β, γ og α representerer rotasjonsvinkler om de respektive aksene

Figur 3.7 viser aksekors i senter av robotens treghetssensor. Tyngdeakselerasjonen vil alltid trekke roboten nedover, dermed kan roll(β) og pitch(γ) vinkler beregnes ved å dekomponere hvor stor andel av tyngdeakselerasjonen som bidrar i sensorens translatoriske retninger. Dette gjøres i åpen-kildekode bibliotek "FreeSixIMU.h" tilhørende IMU sensoren.

Effektsensor

Det montert en energisensor for måling av ytelsesdata under opptrening. Sensoren INA219 [10] overvåker strømtrekket samt spenningsnivået til batteriet. Sensoren måler dermed en sum av all effekt roboten forbruker, inkludert tap i elektronikk og spenningsomformere. Energiforbruk under testing blir beregnet ved å summere opp effekten ganger tidssteget mellom sampler.

Tabell 3.6: INA219 effektsensor

Sensor	INA219
	Batterispenning og strømforbruk
Måleområde	[0-26]V, [0-12.8] A
Grensesnitt	I^2C
Oppløsning	12 bit,23 mW
Samplingsrate	50 Hz
Feilmargin	0.5 %

3.3 Software

Dette delkapittelet består av en overordnet beskrivelse av programkoden til mikrokontrolleren. Programkoden (C++) i sin helhet er vedlagt.

I programkodens hovedfil "main.ino" settes det opp tråder som kjøres ved faste tidsintervaller for å lese inn sensordata, styre servomotorer, håndtere maskinlæring og seriellkommunikasjon, samt oppdatering av display.

Det er laget biblioteker for implementering av Hopf-oscillatorer for lokomosjon og for maskinlæringen. Implementeringen er beskrevet i de to neste delkapitlene.

3.3.1 Implementering av Hopf-oscillatorer

Det er laget et objekt for hvert bein til roboten av klassen "Leg". Objektenes hovedfunksjon er å iterativt beregne Hopf-oscillasjonen som gir motorpådrag til beinene. Klassenes metoder vises i algoritme 1.

Algorithm 1:

```
Klasse: Leg;
Constructor()                                          ▷ Generer initiell oscillatorverdi, ≠ 0
void updateHopfVariables(float A, float B, float b, float wSt, float wSw, float timeStep);
void updateHopfRadie(float Hopf_My)                    ▷ Juster radien for gangen
void CalcHopf();                                       ▷ Kalkuler neste steg i oscillatoren
void GoToZero();                                       ▷ Trekk motorer mot senter
float getHopfX();                                      ▷ Returner oscillatorverdi for motorpådrag
float getHopfY();
```

Bein-objektene beregner settpunkt for servomotorene ved å kalkulere neste steg i differensialligningene til Hopf-oscillatorene. Beregningene blir gjort hvert 20. millisekund, tilsvarende reguleringsfrekvensen til servomotorene. Det kan gjøres sanntids endringer av Hopf oscillatorenes parametere, samt fasekobling av oscillatorene for å endre ganglaget.

X og Y utgang fra hver oscillator styrer henholdsvis kne- og hofte-motorer. Det legges så til forsterkning og faseforskyvning til utgangene. Disse inngår i parametere som det optimaliseres med hensyn på under maskinlæringen. For at kne skal trekkes raskt opp og ned i fasenes endepunkter settes forsterkningen til kne-pådraget veldig høyt, hvor det trekkes opp mot et metningspunkt.

For manuell kjøring av robot via fjernstyring justeres hastigheten for robotens gange ved å definere radien det er ønskelig at oscillatorene skal konvergere mot. Roboten svinger ved å begrense raiden til en av sidene. Beinens gange stanses ved å kalle metoden GoToZero(), hvor motorenes posisjon trekkes mot sine senterposisjoner. Differensialligningene som beregner Hopf-oscillator utgangene er beskrevet i algoritme 2.

Algorithm 2: C++ kode for kalkulasjon av diffligning, Hopf-oscillator

```
funksjon: void CalcHopf();
Constants: τ timestep [float], HopfConstantA, HopfConstantB
r = sqrt(sq(HopfX) + sq(HopfY));
w = Hopf_wSt/(powf(NatNum_E, (−Hopf_b ∗ HopfY)) + 1)+
    Hopf_wSw/(powf(NatNum_E, Hopf_b ∗ HopfY) + 1);
for (int i = 0; i <= 3; i++) {
      sum = sum + GATE[legNumber][i] ∗ GLOBAL_YVAR[i]; }
float xdot = HopfConstant_A ∗ (HopfVar_my − sq(r)) ∗ HopfX − (w ∗ HopfY);
float ydot = HopfConstant_B ∗ (HopfVar_my − sq(r)) ∗ HopfY + (w ∗ HopfX) + sum;
HopfX = HopfX + xdot ∗ timeStep;
HopfY = HopfY + ydot ∗ timeStep;
```

3.3.2 Implementering av treningsalgoritme

For å opprettholde robotens grad av selvstendighet og selvlæring har det vært fokus på å holde treningsalgoritmen internt i roboten. Dermed kan roboten trenes uten styring fra eksterne enheter som datamaskiner eller menneskelig kontakt. Det er derfor utviklet et eget maskinlæringsbibliotek i C++11, hvor de genetiske algoritmene beskrevet i delkapittel 2.4 er implementert.

Initielt blir det laget et objekt av klassen "EvolutionalTrainer", hvor det genereres en populasjon med parametervektorer (kromosom). Vektorene blir tildelt tilfeldige verdier innen deres variasjonsområde.

Algorithm 3: Klasse for maskinlæringsobjekt: EvolutionalTrainer

```
1  Klasse EvolutionalTrainer();
2  Constructor()                                          ▷ Generer initiell populasjon
3  Metoder:
4  void rateChromosome(int populationNr, int fitness);  ▷ Gi ytelsesmål til individ etter endt test
5  evolve();                                              ▷ Utvikle til neste generasjon
6  int getPopulationVariable(int PopNr, int VarNr);
7  int getpopulationFitness(int popNr); int getBestPerformanceVariable(int VarNr);
8  int getBestPerformanceFitness();
9  int getChromosomeSize();
10 int getPopulationSize();
11 int getGeneration();
```

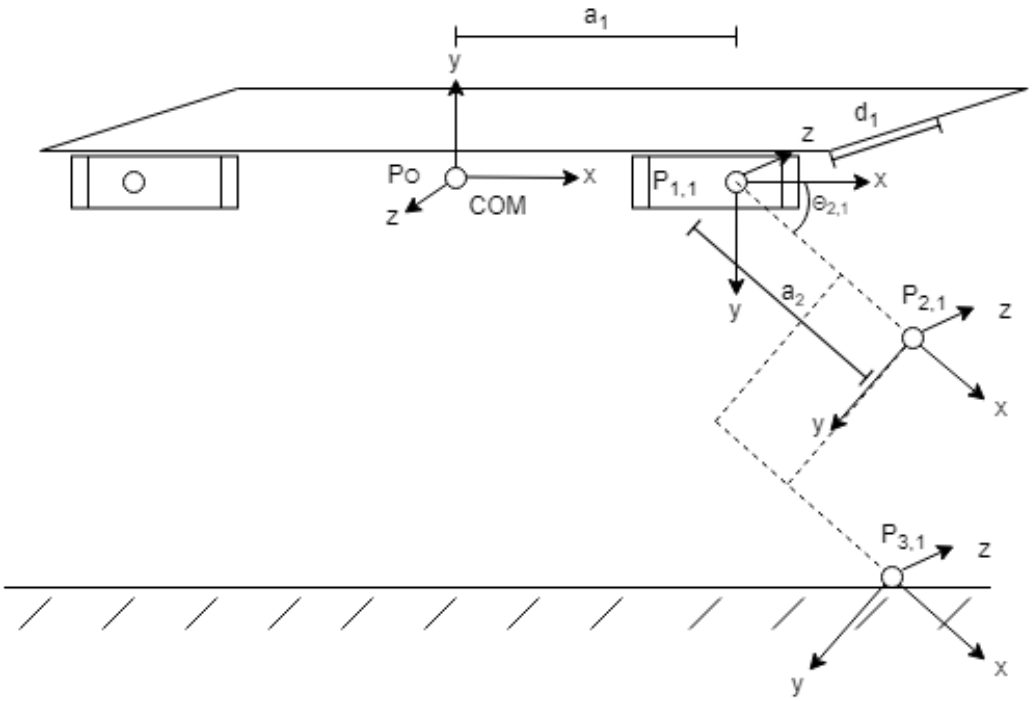
Når roboten er satt i treningsmodus genereres først en populasjon hvor hvert kromosom består av tilfeldig valgte parametere innen variasjonsområde, beskrevet i tabell 3.7. Hver test startes ved å oppdatere robotens noder i nevralnettverket 2.6, og roboten settes igang ved å gi radien til oscillatoren verdi 1. Alle kromosomene blir testet iterativt, hvor de blir rangert ytelsespoeng fortløpende. Når alle parameterne i en generasjon er testet, utvikles en ny generasjon av parameterne ved å kalle metoden evolve(). Populasjonsstørrelse er valgt til 10 for å korte ned på treningstid i hver generasjon. For å unngå at toppunkter i løsningsrommet ikke kan nås på grunn av diskretisering har hvert gen en variasjon på 100.

Tabell 3.7: Genotypen som utvikles i robot-kromosom

Forkortelse	Beskrivelse	Variasjonsområde	Oppløsning	enhet
W_{Pe}	Frekvens, pendelfase	[2, 12]	0.1	$\frac{1}{s}$
W_{Sw}	Frekvens, svingfase	[2, 12]	0.1	$\frac{1}{s}$
G_{Xf}	Forsterkning, hofte bak	[4.5, 49]	0.45	grader
G_{Xb}	Forsterkning, hofte foran	[4.5, 49]	0.45	grader
B_{Xf}	Forskyvning, hofte bak	[-25, 25]	0.5	grader
B_{Yf}	Forskyvning, kne foran	[-15, 15]	0.3	mm
B_{Xb}	Forskyvning, hofte bak	[-25, 25]	0.5	grader
B_{Yb}	Forskyvning, kne foran	[-15, 15]	0.3	mm

3.4 Kinematikk

Det er utledet kinematisk modell som beskriver robotens bein i 3D koordinater gitt vinkelposisjonene til motorene. Hensikten med modellen er å kunne beskrive endeposisjon til føttene gitt det opptrente bevegelsesmønsteret til motorene. Det er kun tatt hensyn til rigid bevegelse uten dynamikk, altså robot-beinenes fot-posisjon gitt motorenes vinkler uten hensyn på dynamikk i ledd, deformasjon, massetreghet, dynamikk i fjær-demper system etc.



Figur 3.8: Figur, aksekors bein 1.
P_o representerer massesenteret. $P_{1,1}$ senter hofte, $P_{2,1}$ senter kne, $P_{3,1}$ kontaktpunkt fot-bakke

Figur 3.8 viser aksekors plassert i robotens ledd i henhold til DH-konvensjonen[3]. Transformasjon fra robotens massesenter P_o til knærne punkt $P_{2,i}$[4] kan da gjøres ved 4 homogene transformasjoner mellom hvert ledd. Tabell 3.8 viser DH-oarameterene for transformasjonene. Grunnet symmetri i designet gjelder parameterne også for de resterende beinene, kun skilt med fortegnendringer.

Tabell 3.8: DH-parametere

Link	d (m)	θ (°)	a (m)	α (°)
1	0.053	180	0.088	0
2	0	$\theta_{2,i}$	0.042	0

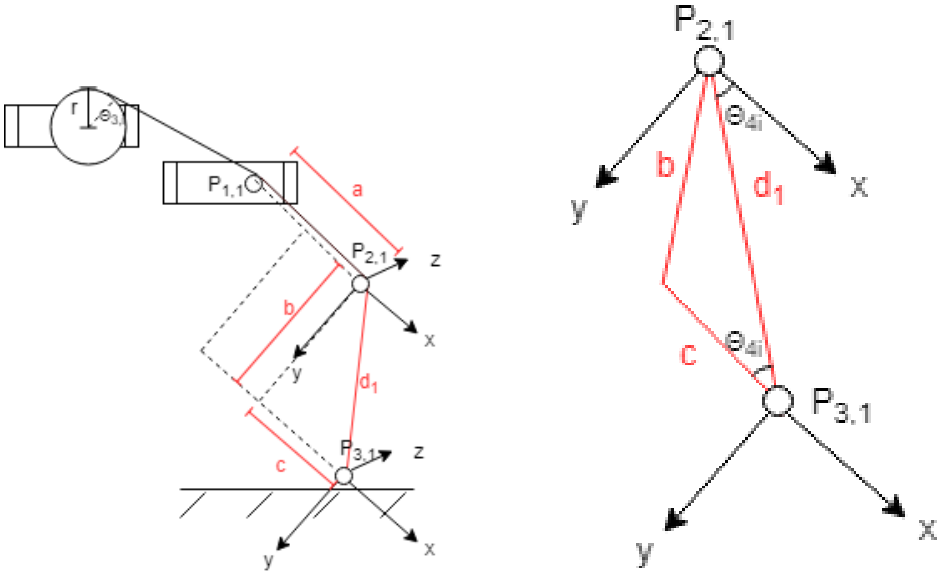
$P3_i = H^2_{3,i}\ H^1_{2,i}\ H^o_{1,i}$

$H = R_{z,\theta}\ Trans_{z,d}\ Trans_{x,a}\ R_{x,a}$

[3]Krav til plassering av aksekors iht. DH-konvensjonen: Aksen x_i må være normalt på og krysse aksen z_{i-1}
[4]i representerer beinnummer, 1-4

Transformasjonen fra kne til fot (punkt $P_{2,i}$ til $P_{3,i}$) endres ved å justere vaierlengde mellom $P_{1,i}$ og $P_{3,i}$. Dette gjøres ved å rotere servomotor, figur 3.9 a).

$$\Delta d_1 = \frac{\Delta\theta_{3,i} * 2 * \pi * r}{360} * (\frac{1}{2})^5$$



(a) Transformasjon, kne til fot

(b) Forenklet

Figur 3.9

Ved å tilnærme at vaieren trekkes igjennom senter av aksekors kan problemstillingen forenkles ned til figur 3.9.

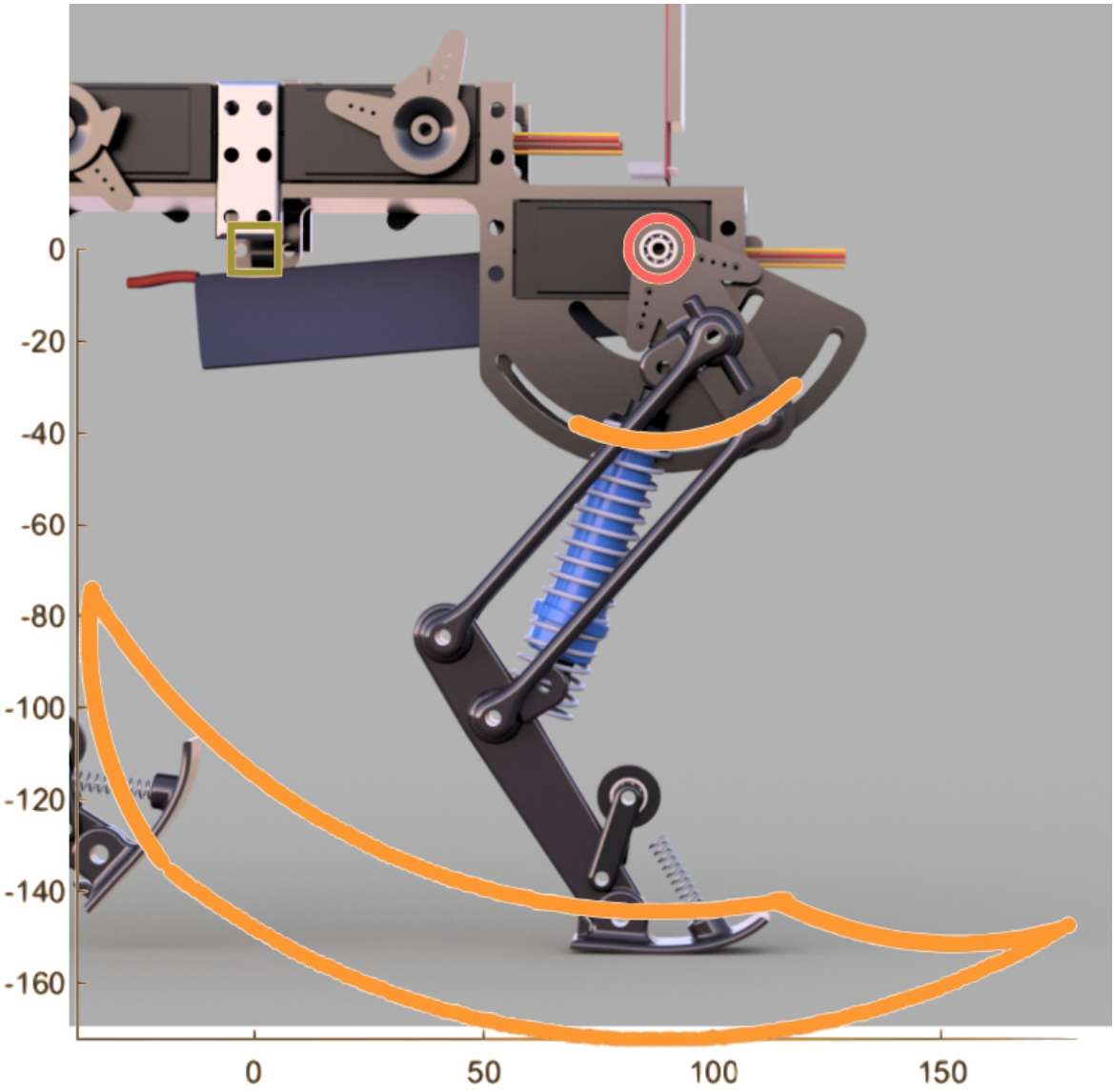
Pantografdesignet sørger for at aksene mellom $P_{2,i}$ til $P_{3,i}$ alltid er parallelle. Transformasjonen er utledet via trigonometri i 3.1 via vinkel θ_{3i}

$$\begin{aligned}
P_{31}, x &= P_{21}, x + d_1\ cos(\theta_4) \\
P_{31}, y &= P_{21}, y + d_1\ sin(\theta_4) \\
\theta_4 &= cos^{-1}(\frac{b^2 + d_1^2 - c^2}{2\ b\ d_1}) \\
P_{31}, x &= P_{21}, x + d_1\ \frac{c^2 + d_1^2 - b^2}{2\ c\ d_1} \\
P_{31}, y &= P_{21}, y + d_1\ \sqrt{1 - (\frac{c^2 + d_1^2 - b^2}{2\ c\ d_1})^2}
\end{aligned} \tag{3.1}$$

Motorene som styrer vinklene til hoftene og sammentrekning av kne er styrt av en sti med settpunkt generert i mikrokontrolleren. (Nærmere beskrevet i delkapittel 2.2).

[5]Giret med trinse i bunn av fot

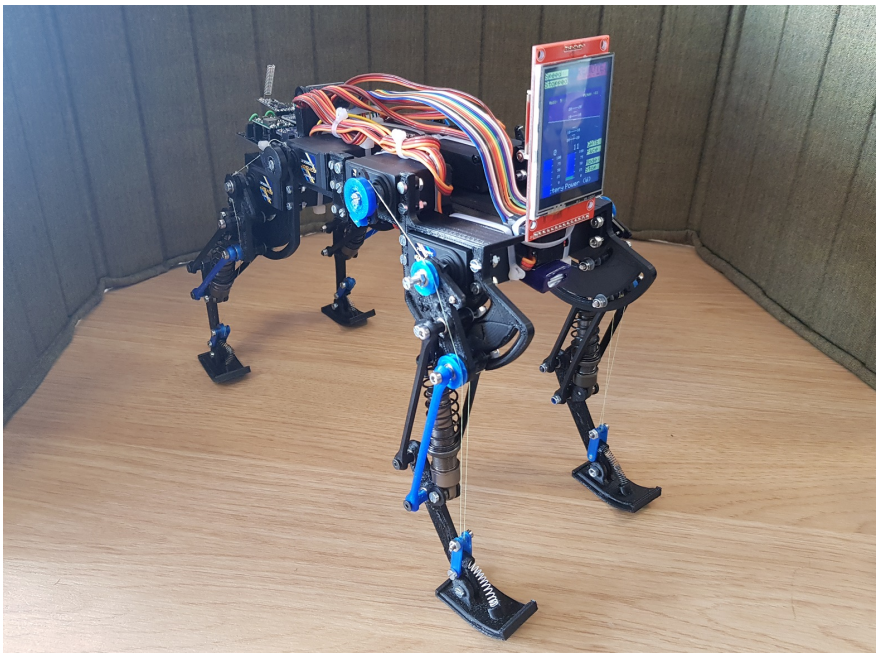
Foroverkinematikken er implementert i MATLAB. Figur 3.10 viser robotføttenes arbeidsområde, overlagt fremstilling av CAD modell.



Figur 3.10: Arbeidsområde til beinene. Massesenteret representert med firkant, rotasjonssenter hofte rød sirkel, orange endeposisjon kne og fot

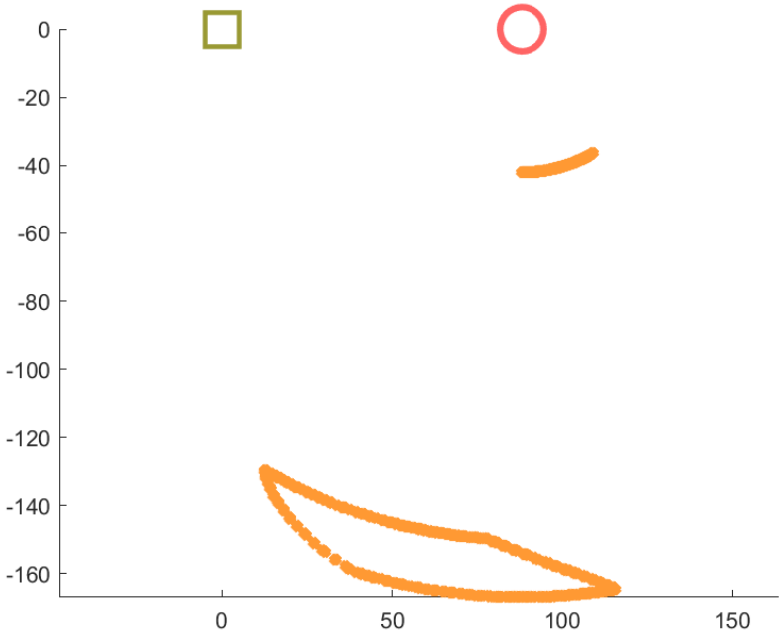
3.5 Oppsummering av firebeinet robotsystem

Det er konstruert en firebeinet robot med 3-segment beindesign3.11. Beinene styres av utgangen fra fire Hopf-oscillatorer koblet i et nevralnettverk. Kombinasjonen av 3 segment bein og Hopf-oscillatorer gir roboten en stabil og robust gange. Enkle settpunkt setter igang lokomosjonsmønsteret. Det er gjort klart 8 noder i et nevralnettverket som skal optimaliseres ved genetiske algoritmer i neste kapittel.



Figur 3.11: Bilde av produsert firebeint robot

Figur 3.12 viser et bevegelsesmønster beinene vil få med noen tilfeldig valgte parametere for nevralnettverket.



Figur 3.12: Bevegelsesmønster for høyre frambein gitt signal fra Hopf-oscillator [mm]
Massesenteret representert med firkant, rotasjonssenter hofte sirkel, oransje endeposisjon kne og fot.

Kapittel 4

Eksperiment

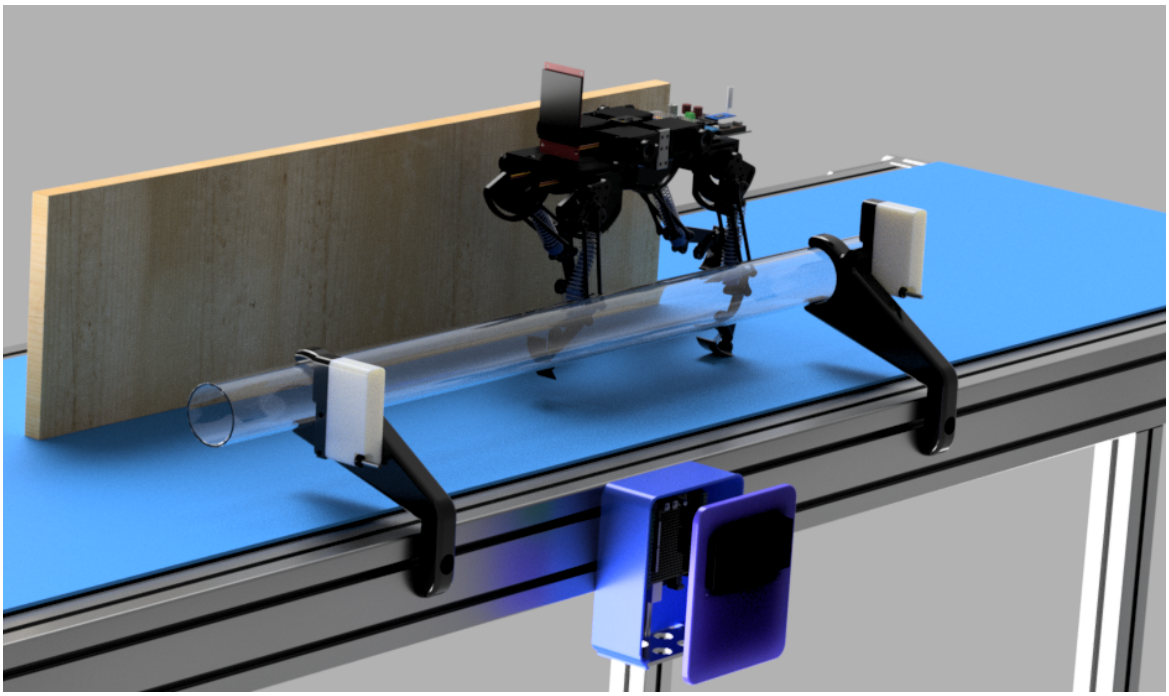
Hensikten med eksperimentet er å utprøve hypotesen definert i innledningskapittelet:

"Kan genetiske algoritmer anvendes for opptrening av sentrale mønstergeneratorer i en firebeint robot?"

Robotens egenskaper for selvlæring utprøves i en testbenk. I opptreningssystemet skal roboten utføre hele optimaliseringsprosessen selvstendig. Sensorer betraktes som robotens sanseorganer, hvor roboten er bevistpå sin egen suksessrate, og skal kunne trene seg opp basert på ytelsesmål. Robotens kromosom som inneholder bevegelsesparameterne utvikles iterativt med evolusjonsalgoritmen genetiske algoritmer.

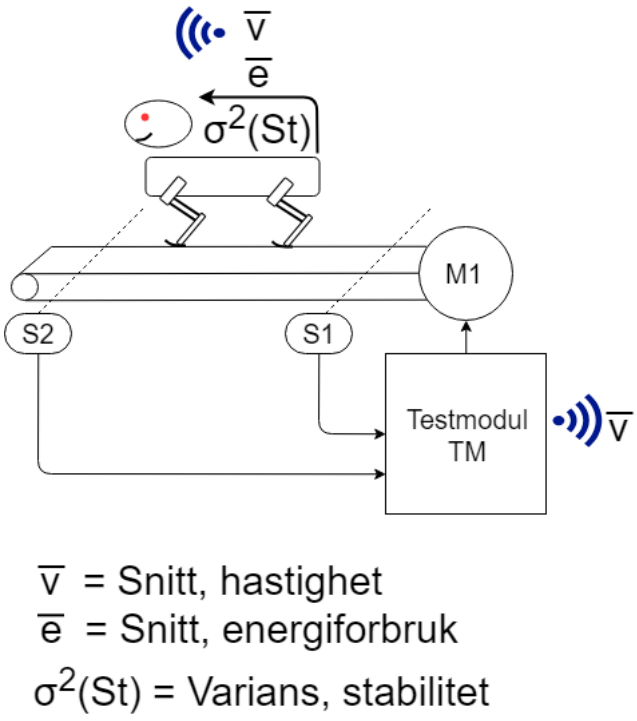
Under opplæringsprosessen trenes bevegelsesparameterne helt uavhengig av ekstern datamaskin eller tilbakemelding fra en testingeniør. Da enkelte parametere gjør så roboten velter eller går bakover kan testen overstyres og avbrytes av en operatør. Ytelsen til kromosomet rangeres da null poeng.

Figur 4.1 viser CAD modell av robotens testoppsett



Figur 4.1: CAD modell av testsystem. Robot på transportbånd med system for tidtaking.

4.1 Testoppsett



Figur 4.2: Testsystem som illustrerer de målbare ytelsene

Selv om roboten er batteridrevet, kjøres testen med tilkoblet ladestrøm slik at utgangspunktet for alle testene skal være mest mulig likt og for å unngå å måtte avbryte tester underveis for å lade roboten.

Gelender er montert på hver side av transportbåndet slik at roboten styres dersom den er på vei ut av treningsområde. Bredden mellom gelenderene er tilstrekkelig slik at roboten ikke skal kunne leneseg inntil gelender under testing.

Hver generasjon består av 10 kromosom, hvor hvert kromosom testes individuelt og blir rangert med ytelsesmål. Hver kjøring er definert som en "kromosomtest".

Testmodulen (TM) kommuniserer trådløst med roboten og har tre funksjoner:

Utvidet sensor

TM klokker tiden det tar fra en test starter til roboten passerer endesensoren. Dermed beregnes robotens gjennomsnittshastighet for hver test.

Styring av transportbånd

Etter roboten har testet et kromosom kjøres transportbåndet bakover for å transportere roboten tilbake til startposisjon. Transportbåndet står i ro når roboten utfører kromosontest.

Datalogger

Treningsinformasjon blir sendt via TM til en datalogger som lager treningsverdier i MATLAB. Forøvrig er TM en slavemodul, all treningslogikk ligger internt i robotens mikrokontroller.

4.1.1 Ytelsesmål

Som utvikler av et robotsystem har vi mulighet til å tilpasse robotens egenskaper ved å velge hvilke ytelsesmål det skal optimaliseres mot. I dette tilfelle er det valgt å undersøke robotens treningsutvikling mot tre forskjellige ytelsesmål: Hastighet, stabilitet og energieffektivitet.

Ytelsesmål og annen relevant data lagres underveis under kromosomtesting. Data fra ytelsesmålene igjennom generasjonene med testing blir fremvist i en læringskurve. Alle ytelsesmål i generasjonene er gjengitt i læringskurven, hvor også gjennomsnittlig ytelsesmål og generasjonens beste er fremhevet. Treningen fortsetter til ytelsene konvergerer, hvor ytelsene ikke lenger forbedres. Det er nødvendigvis ikke alltid progresjon i hver generasjon da noen genkombinasjoner fra suksessfulle foreldre kan ha et negativt utslag. Genmutasjoner kan gi sprang i læringskurven, men grunnet praktiske utfordringer er testene her avsluttet relativt tidlig. Kromosomtesten som gjør det best i løpet av hele testløpet blir lagret i robotens bevegelsesparameter for manuell kjøring.

Det er eksperimentert med å optimalisere robotens bevegelsesparametere ved å de forskjellige ytelsesmålene individuelt. For sammenligning er ytelsesmålene er normalisert til å konvergere mot ca. 200 poeng, hvor økende verdi representerer bedre ytelse.

Hastighet:

$f_v(k) = 254 - t * 10$

Hvor:
t representerer tiden i sekunder fra start av test til passering av endesensoren.

Stabilitet:

$f_s(k) = 254 - (Var(pitch) + Var(roll)) * 5$

Hvor:
Roll og pitch representerer helningsmåling i grader.

Energieffektivitet:

$f_E(k) = 254 - \frac{J}{5}$

Hvor:
J representerer energimål over distansen i Joule.

Energieffektivitet kan videre regnes om til felles enhet COT[1], beskrevet i kapittel. 1.1.1:

$$COT = \frac{E}{mgd}$$

Hvor:

E = Målt energiforbrukt [Joule]
m = 1,036 [kg]
g = 9,81 $[\frac{m}{s^2}]$
d = 0,62 [m]

Alle ytelsesmål er målt ifra roboten står i ro og får startsignal (r = 1), til den passerer endesensoren. På denne måten blir robotens gange evaluert både under igangkjøring og underveis i normal gange. Alle kromosomtester utføres med ganglag type trav.

[1] COT er beregnet fra stillestående start, v = 0.

4.2 Resultat

4.2.1 Manuelt innstilte parametere

For å kunne sammenligne data fra den automatiske maskinlæringen ble det brukt en god time på å stille inn parameterne manuelt. Det ble fokusert på å få roboten til å ha en stabil og rask gange, både basert på visuell tilbakemelding og måledata. Parameterne ble manuelt undersøkt i testbenken på samme måte som under opptreningen for å kunne sammenligne mål av ytelsesdata. Det blir søkt i parameterne ved å analysere hvordan total ytelse endres positivt eller negativt ved endring av parametere, og kan sammenlignes med en gradientsøkalgoritme.

Resultat

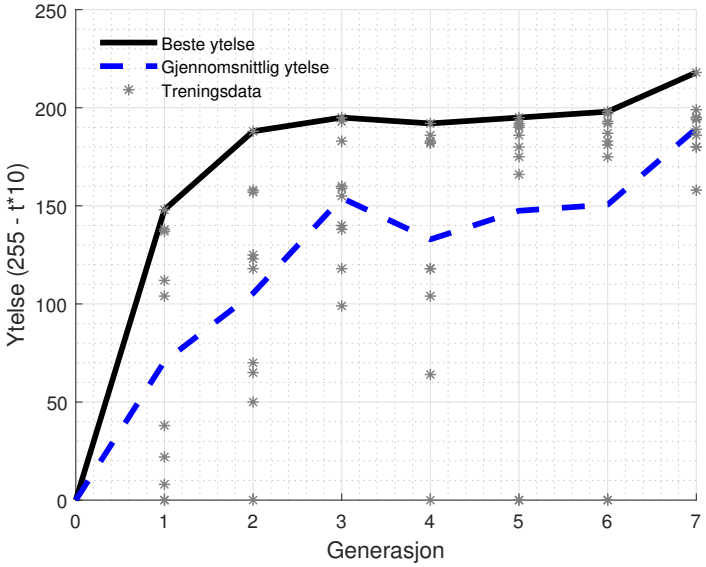
Å trene parameterne manuelt er vanskelig. Parameterne er svært avhengig av hverandre, og det er indikasjoner på at det finnes mange lokale toppunkter i løsningsrommet. Ved de manuelt innstilte parameterne ble laveste energiforbruket målt til 591 J over distansen på 0,62 meter.

Dette tilsvarer en COT verdi på 93,8.

4.2.2 Hastighet som ytelsesmål

Denne testen er utført ved å gi ytelse til roboten kun basert på gjennomsnittlig hastighet. Hastigheten blir målt ved å starte en taimer fra roboten får startsignal, til den passerer endesensor. Totalavstand tilbakelagt er 0,62 m.

Resultat



Figur 4.3: Læringskurve ved ytelse vektet for hastighet.
Sort strek representerer beste ytelse.
Blå strek representerer gjennomsnittlig ytelse.

Testen ble avsluttet etter 7 generasjoner, etter 35 minutter med trening.

Beste gjennomsnittlig hastighet oppnådd $\frac{0{,}62}{3{,}4} = 0{,}18\ \frac{m}{s}$. Dette tilsvarer 0,72 kroppslengder per sekund. Bevegelsesparameterne gitt av kromosomet med best ytelse er vist i tabell 4.1.

Ved å anvende hastighet som ytelsesmål konvergerer roboten raskt. Roboten ender opp med parametere som gir lange og raske steg, hvor robotens stabilitet ansees som dårlig. Roboten hviler litt mot rekkverket når den går på grunn av store svingninger i pitch akse.

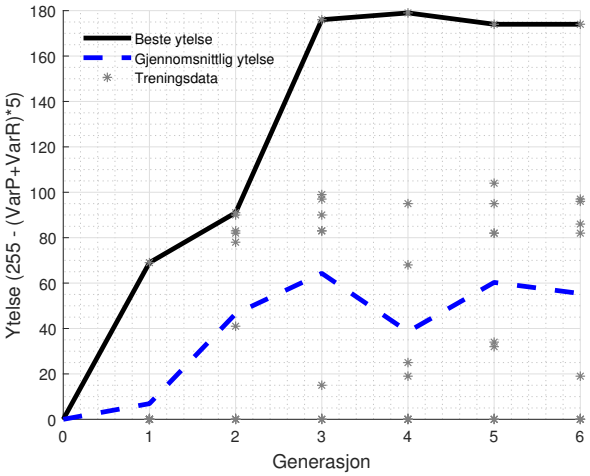
Tabell 4.1: Bevegelsesparametere fra kromosom med best ytelse

Parameter	W_{Pe}	W_{Sw}	G_{Xf}	G_{Xb}	B_{Xf}	B_{Yf}	B_{Xb}	B_{Yb}
Enhet	$\frac{1}{s}$	$\frac{1}{s}$	grader	grader	grader	mm	grader	mm
Verdi	10	10	36	9	-10	9	0	-9

4.2.3 Stabilitet som ytelsesmål

I denne testen utvikles kromosomene basert på stabilitet. Varians i roll og pitch retning blir målt fra roboten får startsignal til den har passert endesensor. Dersom roboten ikke passerer endesensor i løpet av 25 sekunder gis kromosomtesten 0 ytelsespoeng.

Resultat

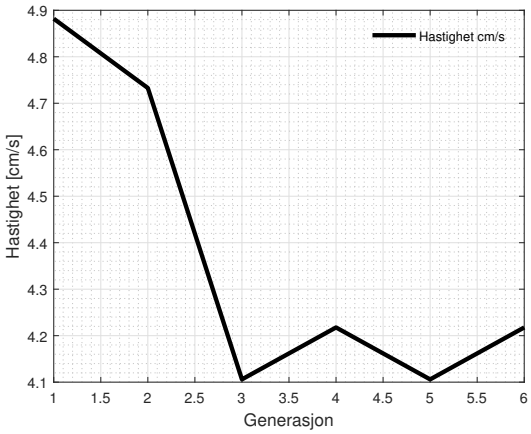


Figur 4.4: Læringskurve ved ytelse vektet for stabilitet.
Sort strek representerer beste ytelse, blå strek representerer gjennomsnittlig ytelse.

Testen avsluttes etter 6 generasjoner, etter 30 minutter med trening. Kromosomet med best ytelse blir funnet i generasjon 4. Parametere er gjengitt i tabell 4.2.

$Var(Roll) = 13 \qquad Var(Pitch) = 2 \qquad f_s(k) = 179$

Roboten utvikler gange hvor overkroppen er veldig stødig. Det ble observert at roboten holdt lav hastighet, og roboten saktet ned igjennom generasjonene. Dette ble bekreftet ved å analysere hvordan hastigheten utviklet seg, vist i figur 4.5.



Figur 4.5: Utvikling av hastighet for kromosonet med generasjonens beste ytelse.

Tabell 4.2: Bevegelsesparametere fra kromosom med best ytelse

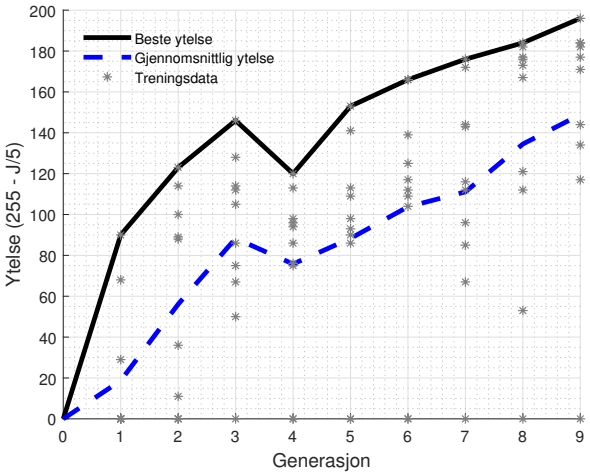
Parameter	W_{Pe}	W_{Sw}	G_{Xf}	G_{Xb}	B_{Xf}	B_{Yf}	B_{Xb}	B_{Yb}
Enhet	$\frac{1}{s}$	$\frac{1}{s}$	grader	grader	grader	mm	grader	mm
Verdi	2	8	36	9	0	-15	-10	-9

4.2.4 Energieffektivitet som ytelsesmål

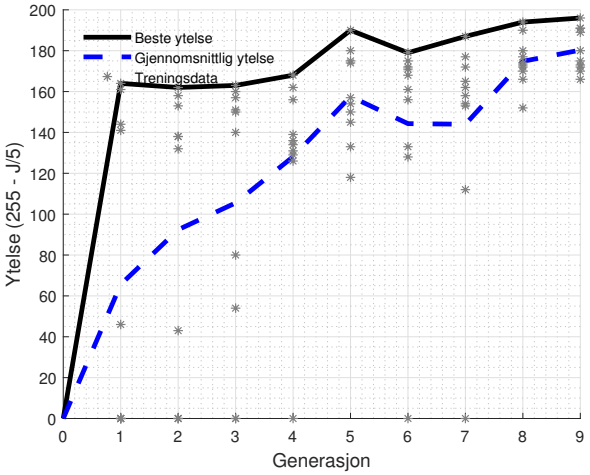
Ved denne testen trenes robotens gange ved å anvende energiforbruk som ytelse. Energiforbruk i Joule blir beregnet fra startsignal til roboten har passert endesensor. Dersom roboten ikke passerer endesensor i løpet av 25 sekunder gis kromosomtesten 0 ytelsespoeng.

Resultat

Det er utført to tester, hvor resultatene vises i figur 4.6 og 4.7. Testene er avsluttet etter 9 generasjoner, ca tre kvarters trening hver.



Figur 4.6: Energitest 1
Læringskurve ved ytelse vektet for energiforbruk.
Sort strek representerer generasjonens beste ytelse.
Blå strek representerer gjennomsnittlig ytelse.



Figur 4.7: Energitest 2
Læringskurve ved ytelse vektet for energiforbruk.
Sort strek representerer generasjonens beste ytelse.
Blå strek representerer gjennomsnittlig ytelse.

Roboten har her tydelig forbedring gjennom generasjonene. Beste ytelse er målt i "Energitest 1", generasjon 9. Kromosomet med best ytelse forbrukt 290 Joule på å gå distansen på 0.62 meter. Forøvrig har roboten en rask og stabil gange.

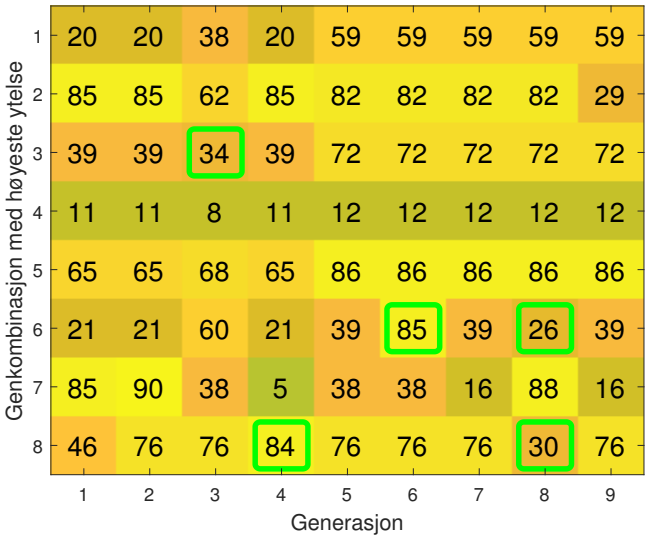
Dette tilsvarer en COT verdi på 46,02. Til sammenligning er dette en reduksjon på 50,9 % sammenlignet med manuelt innstilte parametere.

4.2.5 Videre analyse av treningen med best resultat

Etter å ha studert testresultatene er det gjort avgjørelse av hvilken test som har vist til best treningsresultater. Dette er basert på en subjektiv bedømmelse av robotens gange gitt utviklingen under trening. Det er funnet at "Energitest 2" er den som kommer best ut, representert med læringskurve 4.7. Dette delkapitlet gjør en dypere analyse av treningsdata fra testen for å kunne ta et endelig valg av robotens bevegelsesparametere.

Kromosomutvikling

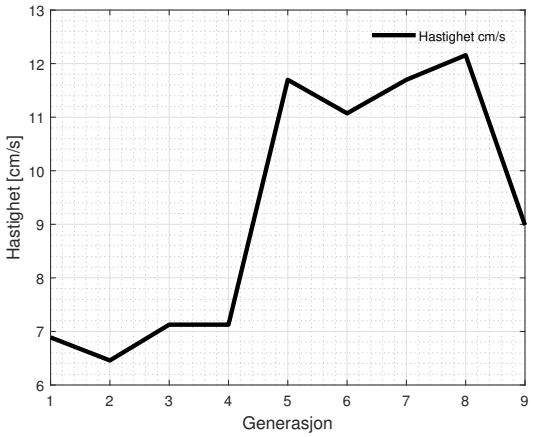
Det er interessant å studere hvordan kromosomene har utviklet seg igjennom testens generasjoner. Kromosomet med best ytelse[2] i hver generasjon er gjenvist i gridkart figur 4.8. Gen tilkommet via mutasjon er representert med grønn firkant.



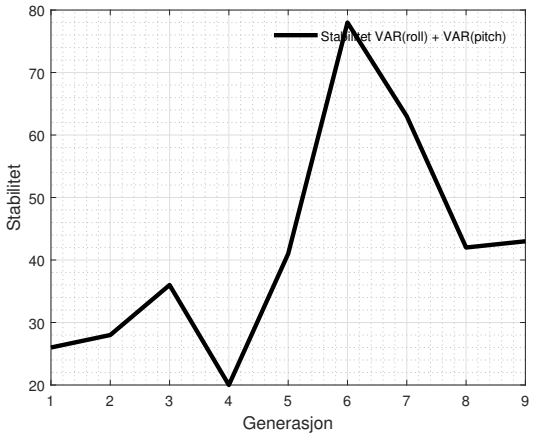
Figur 4.8: Utvikling av kromosom med høyest ytelse i populasjonen
Grønn firkant representerer gen som har tilkommet via mutasjon.

Utvikling, hastighet og stabilitet

Figur 4.9 a) og b) viser utvikling av henholdsvis hastigheten og stabilitet tilhørende generasjonenes beste kromosom.



(a) Utvikling av hastigheten (cm/s) gitt kromosom med høyest ytelse i generasjonene



(b) Utvikling av stabilitet (VAR(roll) + VAR(pitch)) gitt kromosom med høyest ytelse i generasjonene.

Figur 4.9

[2]Hvert gen er representert i heltall fra 0 til 99. Beregning til bevegelsesparametere kan gjøres via tabell 3.7 side 30.

Valg av endelige parameter for robot

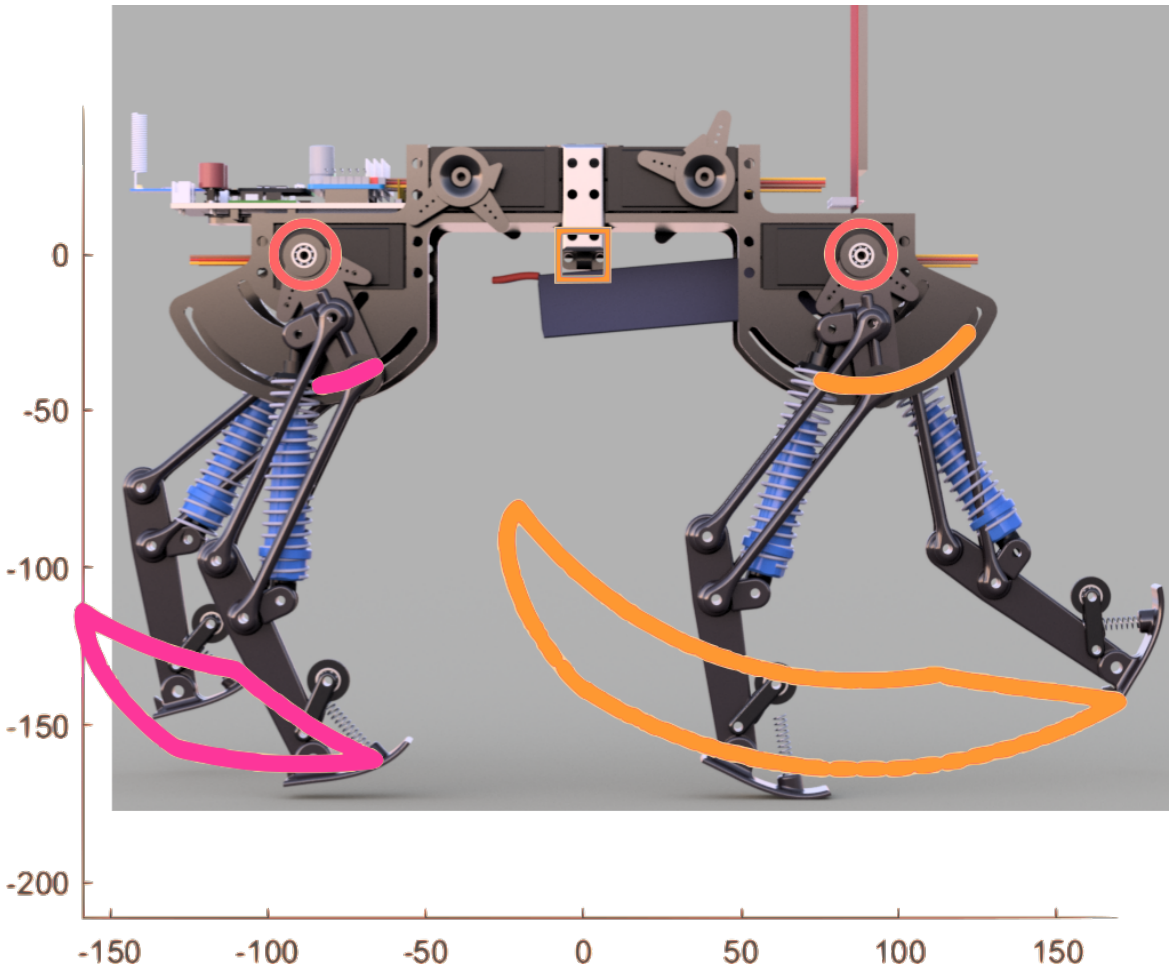
Basert på analysen velges det beste kromosomet i fra generasjon 8. Disse parameterne er en av de mest energieffektive, i tillegg til at hastigheten er relativt høy, og stabiliteten er god. Disse blir programmert som robotens endelige bevegelsesparametere.

De best trente parameterne er representert i tabell 4.3.

Tabell 4.3: Bevegelsesparametere fra kromosom med best ytelse

Parameter	W_{Pe}	W_{Sw}	G_{Xf}	G_{Xb}	B_{Xf}	B_{Yf}	B_{Xb}	B_{Yb}
Enhet	$\frac{1}{s}$	$\frac{1}{s}$	grader	grader	grader	mm	grader	mm
Verdi	7,9	10,2	36,9	9,9	18	-7,2	19	-6

Figur 4.10: Robotens opptrente bevegelsesmønster utregnet via foroverkinematikk

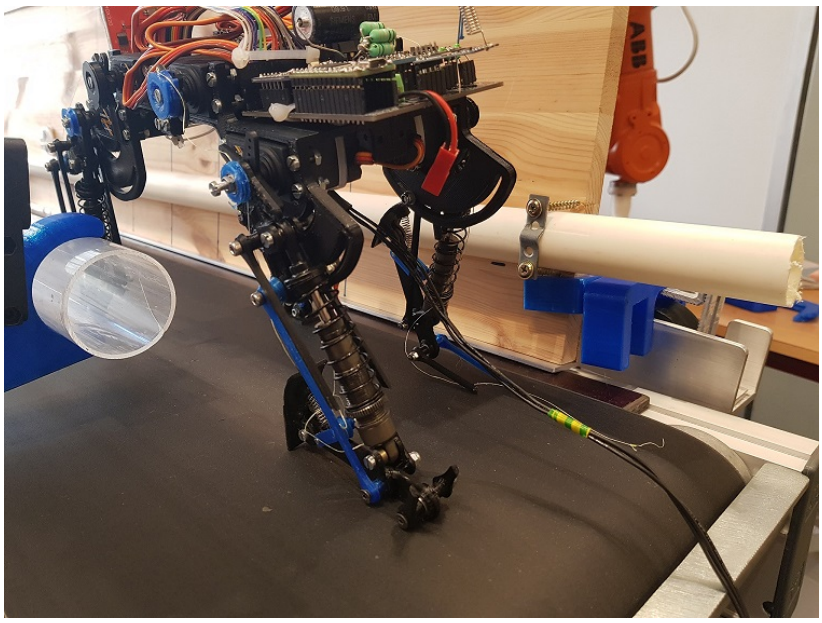


Figur 4.11: Bevegelsesmønster for bein på høyre side, gitt av Hopf- oscillatorer med de optimale trente parameterne for økt energieffektivitet. Massesenteret representert med firkant, rotasjonssenter av hofte sirkel, orange og rød: endeposisjon kne og fot

4.2.6 Test av adaptive egenskaper

Hensikten med denne testen er å undersøke robotens evne til å tilpasse en signifikant endring av robotens interaksjon med omgivelsene. En kan se for seg en situasjon hvor roboten har fått ødelagt to bein i en utilgjengelig omgivelse, og roboten er overlatt til seg selv. Bilde 4.12 viser roboten med knekte parallellstag til beinene.

De mest effektive parameterne fra 4.2.5 gir nå en ustabil gange hvor roboten ikke klarer å bevege seg framover.

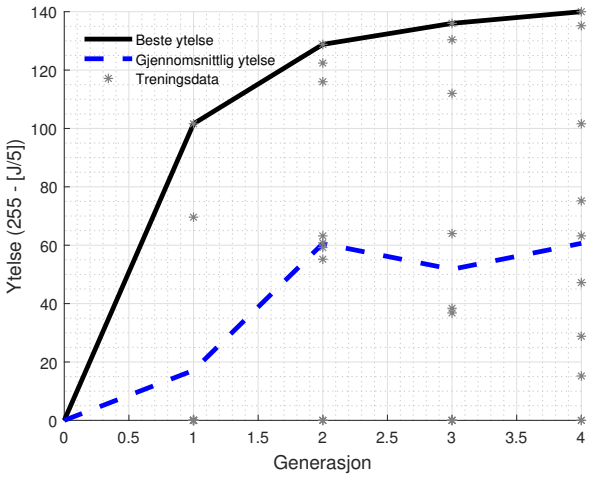


Figur 4.12: Robot under opptrening med beinbrudd

Det blir startet en ny treningsprosedyre med likt testoppsett som ved trening med ytelse som energimål, delkapittle 4.2.4. Figur 4.13 viser treningskurven under opptreningen.

Resultat

Av de initielle parameterne i den første populasjonen er det kun tre av ti kromosomer som fører til at roboten klarer å nå mållinjen. Roboten sliter tydelig med å gå i de nye omgivelsene. Etter kun to generasjoner med testing har roboten funnet tilstrekkelig gode parametere. Roboten halter mer i gangen, men finner parametere den klarer å gå med.



Figur 4.13: Læringskurve, trening med knekte bein

Kapittel 5

Diskusjon og konklusjon

Arbeidet i denne oppgaven presenterer en kombinasjon av ulike biologiske inspirerte metoder innen et robotsystem. I introduksjonen ble det stilt en hypotese:

"Kan genetiske algoritmer anvendes for opptrening av sentrale mønstergeneratorer i en firebeint robot?"

For å besvare spørsmålet, analyserer vi først de tre hovedelementene hver for seg.

5.1 3D printet robot med biologi-inspirert konstruksjon

Anvendelse av 3D printer i en prototype-fase har vært essensielt for å kunne utvikle roboten under en kort tidsperiode ved prosjektets oppstart. Kun timer etter en idé er tegnet kan deler settes sammen og systemer prøves ut.

Ved å lage bein i et 3-segment-design har roboten et fleksibelt og robust ganglag. Løsningen har også gjort det mulig å flytte motorene opp fra beinet, slik at motorene ikke gir ekstra treghetsmoment ved sving av beinene.

Hvert bein har kun to frihetsgrader for lokomosjon, Dette begrenser robotens muligheter for lokomosjon ved ujamnt terreng eller for balansekorrigering. Bruksområde til roboten er derav begrenset til de forholdene den er opptrent til, nemlig flatt gulv. Ved å implementere enda en frihetsgrad i beinene ville det åpnet seg flere muligheter for en mer kompleks lokomosjon.

5.2 Hopf-oscillator for bevegelseskontroll

Hopf-oscillatorer kan anvendes for etterligning av sentrale mønstergeneratorer, og er en adaptiv og robust metode for lokomosjon i robotapplikasjoner. Få og konstante pådragssignaler styrer oscillasjoner som representerer egenskaper for god lokomosjon. Oscillatorene har alltid kontinuerlig transisjon mellom hastighetsendring, igangkjøring og ved endring av bevegelsessammenheng mellom gangmønster.

5.3 Genetiske algoritmer for maskinlæring

Resultatene fra treningen viser at maskinlæring ved GA kan brukes i opptrening av bevegelsesparameterne for firebeint robot. Det ble eksperimentert med ulike valg av ytelsesmål, hvor roboten hadde en positiv læringskurve ved alle tre ytelsesmålene. I denne problemstillingen er det funnet at energieffektivitet er det ytelsesmålet som gir roboten den beste læringskurven. Roboten forbedret seg igjennom flere generasjoner med trening, hvor robotens ferdige opptrente gange kan beskrives som stabil, rask og effektiv.

Av praktiske årsaker er testene avsluttet tidlig. Analyse av resultat viser at de fleste parameterne det trenes mot er "de beste av de initielle parametere" i den første populasjonen. Ved å øke treningstiden er ville flere parameter blitt introdusert via genmutasjoner, som videre kunne ført til sprang i læringskurven.

Anvendelse av GA i en praktisk applikasjon har gitt lovende resultater. Men metoden er begrenset til de prosesser som tåler at parametere blir eksperimentert med i alle arbeidsområder. For slike prosesser kan det være en løsning å simulere problemstillingen.

Resultatene viser at maskinlæring med GA har forbedret robotens effektivitet 50.9%, ned til COT = 46,02 E/(m g d). For sammenligning kan det nevnes at mus har en COT verdi på ca 42, mens hunder har COT verdi ca 1,3.

Eksperiment har også vist at GA kan anvendes til å lære en robot å gå i ukjente omgivelser.

5.4 Konklusjon

- Eksperiment viser at GA kan anvendes til maskinlæring for en praktisk robotapplikasjon. Resultatene viser at opptrening av bevegelsesparameterne har forbedret energieffektiviteten med over 50 % sammenlignet med manuelt innstilte parametere. Dermed kan roboten kan gå over en betraktelig lenger distanse mellom batterilading.
- Å definere gode, målbare ytelsesmål er en meget viktig del av GA. Her ble det eksperimentert med tre forskjellige ytelsesmål, hvor robotens gange endte opp svært forskjellig basert på hvilke ytelser som ble vektet.
- Ytelsesmål basert på energieffektivitet ga den treningskurven med tydelig progresjon igjennom flere generasjoner. Dette ytelsesmålet ga også stabil og rask gange.
- I en praktisk applikasjon er begrensningene av antall løsninger i optimaliseringsproblemet essensielt. Her er det valgt å implementere Hopf oscillatorer i et nevralnettverk, hvor 8 noder i nettverket optimaliseres.
- Oppgaven presenterer en løsning hvor GA typisk kan anvendes som en byggeblokk for opptrening i en uforutsett omgivelse i et autonomt system. En robot kan unngå å få en "deadlock" situasjon ved å trene seg opp til å mestre problemer den støter på.

Etter arbeid med denne oppgaven har jeg en generell oppfatning at biorobotikk og evolusjonsalgoritmer er en god løsning for utvikling av robotapplikasjoner. Det anbefales å anvende systemkomponenter inspirert av biologi for selvlæring og bevegelseskontroll for å drive forskningen videre. Dermed kan vi som utviklere gi ifra oss noe av detaljstyringen, og heller undersøke hvordan vi kan konstruere robotsystemer som anvender maskinlæring for opptrening på et overordnet nivå. Her er det dog viktig at med et stort fokus på etiske rammer.

5.5 Videre arbeid

Under arbeidet med oppgaven er det oppdaget flere områder det kan være interessant med videre utforskning. Spesielt innen maskinlæring ved genetiske algoritmer og implementering av sentrale mønstergeneratorer i roboter.

For videre utvikling av mobile roboter foreslåes følgende:

- Implementere flere oscillatorer i nevralnettverket for mer avansert bevegelsesmønster.
- Implementere sanntids sensordata ved noder i nevralnettverket for styring av robotens balanse, svinging, baneplanlegging og manøvrerering over ujevnt terreng.
- Det ble valgt å ikke simulere roboten i denne oppgaven. Ved simulering av problemstillingen kan det utforskes hvorvidt optimale bevegelsesparameter finnes ved lengre opptreningstid og større populasjon under optimaliseringen med GA.
- Arbeidet med sentrale mønstergeneratorer kan implementeres i andre applikasjoner. Eksempelvis kan CPG nettverk for å kontrollere aktuatorer for vinger i UAV lignende fugler. Her hadde det vært spennende å anvende GA for å optimalisere mhp. energieffektivitet (COT), akustisk støy eller hastighet.

Det er laget en video for demonstrasjon av roboten og opplæringsprosessen, se lenke:

`https://youtu.be/zNXgT2csQ7A`

Bibliografi

[1] V A Tucker. The energy cost of moving about. 63:413–9, 07 1975.

[2] Nicolas Bredeche, Jean-Marc Montanier, Wenguo Liu, and Alan F.T. Winfield. Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):101–129, 2012.

[3] Key Gew Chae and Jong Hyeon Park. Trajectory optimization with ga and control for quadruped robots. *Journal of Mechanical Science and Technology*, 23(1):114–123, Jan 2009.

[4] Peter Eckert, Alexander Spröwitz, Hartmut Witte, and Auke Ijspeert. Comparing the effect of different spine and leg designs for a small bounding quadruped robot. *Proceedings of ICRA 2015*, pages 3128–3133, 2015.

[5] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim. Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 1028–1035, Nov 2015.

[6] Martin Fischer and Reinhard Blickhan. The tri-segmented limbs of therian mammals: kinematics, dynamics, and self-stabilization–a review. *Journal of experimental zoology. Part A, Comparative experimental biology*, 305 11:935–52, 2006.

[7] Lena Mariann Garder and Mats Erling Høvin. Robot gaits evolved by combining genetic algorithms and binary hill climbing, 2006.

[8] J. H. Holland. Adaptation in natural and artificial systems. *University of Michigan Press.MI: University of Michigan Press*, 01 1975.

[9] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642 – 653, 2008. Robotics and Neuroscience.

[10] Texas Instruments. Ina219. `http://www.ti.com/lit/ds/symlink/ina219.pdf`, 2015. Datasheet Zero-Drift, Bidirectional Current/Power Monitor With I 2C Interface.

[11] M. Kelly, M. Sheen, and A. Ruina. Off-line controller design for reliable walking of ranger. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1567–1572, May 2016.

[12] M Anthony Lewis, Andrew H Fagg, and George A Bekey. Genetic algorithms for gait synthesis in a hexapod robot. In *Recent trends in mobile robots*, pages 317–331. World Scientific, 1993.

[13] J. D. Lohn, D. S. Linden, G. S. Hornby, A. Rodriguez-Arroyo, S. E. Seufert, B. Blevins, and T. Greenling. Evolutionary design of a single-wire circularly-polarized x-band antenna for nasa's space technology 5 mission. In *2005 IEEE Antennas and Propagation Society International Symposium*, volume 2B, pages 267–270 vol. 2B, July 2005.

[14] Eve Marder and Dirk Bucher. Central pattern generators and the control of rhythmic movements. *Current biology*, 11(23):R986–R996, 2001.

[15] Jerrold E Marsden and Marjorie McCracken. *The Hopf bifurcation and its applications*, volume 19. Springer-Verlag New York Heidelberg Berlin, 1976.

[16] Julien Nicolas. Artificial evolution of controllers based on non-linear oscillators for bipedal locomotion. *Master's thesis, EPFL, winter*, 2005.

[17] Miguel Oliveira, Cristina Santos, Manuel Ferreira, Lino Costa, and Ana Rocha. Locomotion gait optimization for a quadruped robot. 05 2018.

[18] L. Righetti and Auke Jan Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1585–1590, May 2006.

[19] S. Rutishauser, A. Sprowitz, L. Righetti, and A. J. Ijspeert. Passive compliant quadruped robot using central pattern generators for locomotion control. In *2008 2nd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 710–715, Oct 2008.

[20] Sangok Seok, Albert Wang, Meng Yee Chuah, David Otten, Jeffrey H. Lang, and Sangbae Kim. Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot. *2013 IEEE International Conference on Robotics and Automation*, pages 3307–3312, 2013.

[21] Alexander Sprowitz, Max Fremerey, Konstantinos Karakasiliotis, Simon Rutishauser, Ludovic Righetti, and A.J. Ijspeert. Compliant leg design for a quadruped robot. 01 2009.

[22] Alexander Spröwitz, Alexandre Tuleu, Massimo Vespignani, Mostafa Ajallooeian, Emilie Badri, and Auke Jan Ijspeert. Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot. *The International Journal of Robotics Research*, 32(8):932–950, 2013.

[23] K. S. Tang, K. F. Man, S. Kwong, and Q. He. Genetic algorithms and their applications. *IEEE Signal Processing Magazine*, 13(6):22–37, Nov 1996.

Kapittel 7

Vedlegg

Type	Navn	Beskrivelse
Datablad	Teensy 3.2	Mikrokontroller
	MP158	5V step down DCDC
	10 DOF IMU	Treghetssensor
	HC-12	Trådløs modul
	INA219	Effektsensor
	10 PCA9685	PWM driver
Programkode C++	main	Hovedprogram
	Display	Bibliotek, skjerm
	EvolutionalTrainer	Bibliotek, maskinlærings
	Leg	Bibliotek, Hopf-oscillatorer
CAD filer for 3D print		Robotens deler i .cmf format

Tabell 7.1: Tabell over vedlegg.

Vedlegg er innbakt i pdf filen i .7zip format.