Aalto-yliopisto
Perustieteiden korkeakoulu
Teknillisen fysiikan ja matematiikan tutkinto-ohjelma

Näyttämällä opettaminen: liikeratojen toistamisen mallit

erikoistyö
17.10.2013

Lasse Lindqvist

Valvoja: Harri Ehtamo
Ohjaaja: Harri Ehtamo

Sisältö

1 Johdanto

Näyttämällä opettaminen tarkoittaa robotiikassa tapaa opettaa robotille liike- tai toimintasarja siten, että käyttäjä ensin näyttää itse, kuinka toiminta tehdään.[12] Havainnollistaminen voidaan tehdä joko liikuttamalla robotin liikkuvia osia tai näyttämällä itse omalla esimerkillään toiminnan. Tämä on hyvin toteutettuna selvästi tehokkaampi tapa opettaa robotteja kuin esimerkiksi koordinaattipisteiden ja vastaavien syöttäminen käsin järjestelmään.

Tässä työssä keskitytään siihen, miten robotti voi oppia liikkeen saatuaan esimerkkitoiminnan selville esimerkiksi kuvattuaan opettavan ihmisen liikesarjan tai tallennettuaan omien niveltensä liikkeen. Työssä ei puututa robotteihin sinänsä, vaan tarkastellaan asiaa matemaattisen mallin rakentamisen kannalta. Robotiikka tuo mukanaan lisähaasteen toimintojen oikeaan toistamiseen muun muassa fysikaalisten rajoitusten takia.

Työssä käydään läpi liike- tai toimintasarjojen toistamiseen käytettävien mallien ominaisuuksia ja toteutustapoja. Tällaisia ovat esimerkiksi ajan ottaminen huomioon toistossa, stabiilisuus ja vaadittavan laskennan kompleksisuus. Ajan huomioiminen esimerkiksi tuottaa samalla mahdollisuuksia erilaisiin hidastamis- tai kiinniottoliikkeisiin, mutta monimutkaistaa toimintaa häiriötilanteissa.

Verrattavana on kolme erilaista tapaa mallintaa toimintasarjoja. Näiden mallien ominaisuudet käydään vertaillen läpi. Ominaisuuksia esitellään myös erilaisten esimerkkiliikeratojen avulla. Malleista on olemassa eri versioita ja nämä esitetään vertaillen.

Lopuksi esitetään yhteenveto eri malleista ja pohditaan minkälaisiin tilanteisiin tai järjestelmiin eri mallit voisivat parhaiten sopia. Malleille esitellään laajennusehdotuksia erinäisiä ominaisuuksia varten. Lisäksi esitellään lyhyesti muutama vaihtoehtoinen malli.

2 Liikeradan toistaminen ja robotiikka

Liikeradan toistaminen dynaamisessa systeemissä tarkoittaa yksinkertaisimmillaan sitä, että syötteenä on paikkakoordinaatteja eri ajanhetkillä ja näistä lasketaan kiihtyvyydet, joiden perusteella liikerata toistetaan. Tämä ei kuitenkaan ole mielekästä tai tehokasta, sillä luotu malli pätee vain tietyssä tapauksessa. Tällainen malli ei osaa käsitellä minkäänlaisia häiriöitä tai muutoksia ohjattavan objektin tai maalin sijainnissa. Tämän takia on tarkoituksenmukaista löytää malli, joka sisältää jonkinlaisen ohjauksen.

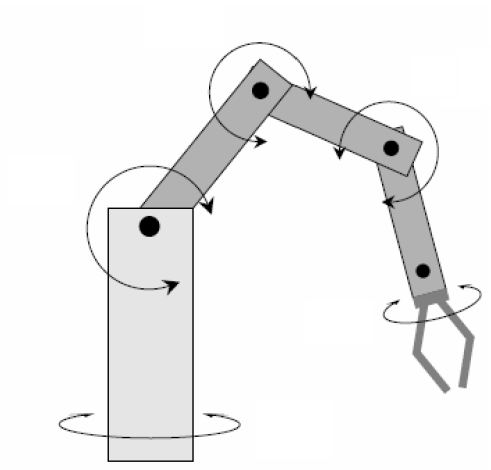
Mallin syöte, eli liikerata, koostuu tavallisesti tasavälein poimituista koordinaateista. Nämä koordinaatit voivat olla esimerkiksi kolme paikkakoordinaattia, jolloin käytössä on vain paikka avaruudessa ilman suuntausta tai jotakin pintaa vasten kohdennettua voimaa. Vaihtoehtoisesti koordinaatit voivat olla nivelten asentoja. Kuvassa 1 on esimerkki robottikädestä. Tässä tapauksessa robottikäden pihtien paikka voidaan ilmoittaa kolme paikkakoordinaatin ja kolmen suuntauskoordinaatin avulla tai vaihtoehtoisesti käden viiden nivelen kulmien avulla. Tässä työssä ei käsitellä suuntauksia tai voimia, sillä esitettävät mallit voidaan tutkia riittävästi pelkästään paikkakoordinaattien kanssa.

Roboteissa käytetään yleisesti suoraa kinematiikkaa, jolla robotin nivelten asennoista voidaan laskea paikka ja suuntaukset kolmiulotteisessa avaruudessa. Samoin käytetään käänteiskinematiikkaa, kun lasketaan mahdolliset nivelten asennot annetusta kolmiulotteisen avaruuden pisteestä ja suuntauksesta.[13] Analyyttinen ratkaisu käänteiskinematiikkaongelmaan on mahdollista löytää vain viiden vapausasteen ongelmille ja kuuden vapausasteen erityistapauksille, minkä takia usein käytännössä robottikädet rakennetaan näitä rajoituksia noudattaen.[3]

3 Mallin ominaisuudet

3.1 Koordinaatiston valinta

Sillä, käytetäänkö liikeradan toistamisessa kolmiulotteista koordinaatistoa vai robottikäden omia nivelten asentoja, voi olla merkitystä suorituksen kannalta. Reaalikoordinaatistossa liikkeet ovat useimmin sileitä. Nivelmaailmassa liikkeet sen sijaan ovat tavallisesti derivaataltaan epäjatkuvia, sillä nivel kääntyy tavallisesti vain kahteen suuntaan ja kesken liikkeen yksittäinen ni-

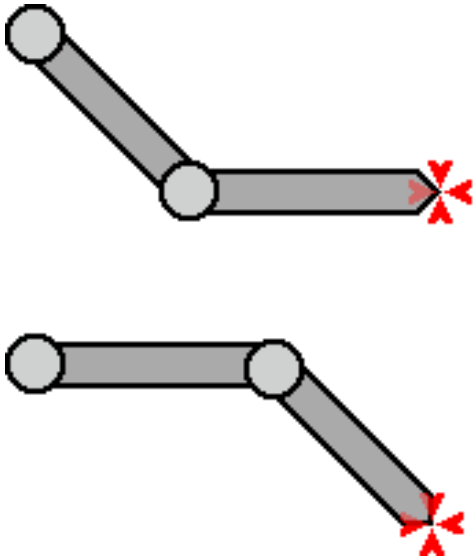


Kuva 1: Tällä robottikädellä on viisi vapausastetta. Pihtien paikka voidaan ilmoittaa kolmen paikkakoordinaatin ja kolmen suuntauskoordinaatin avulla tai vaihtoehtoisesti eri nivelten kulmien avulla.

vel voi vaihtaa liikesuuntaansa päinvastaiseksi. Nivelavaruudessa operoidessa pitää siis pitää huolta, että valittu malli pystyy käsittelemään nopeat suunnanvaihdokset. Toisaalta reaalikoordinaatistossa on mahdollista käydä niin, että kun laskettu tavoitepiste eroaa esimerkkipisteestä tarpeeksi esimerkiksi laskentaepätarkkuuksien takia, päädytään hyvinkin erilaisiin nivelten asentoihin. Tämä voi aiheuttaa muun muassa törmäysvaaran tai yrityksen tehdä mahdottomia liikkeitä. Kuvassa 2 on esitetty, kuinka sama piste voidaan saavuttaa usealla eri tavalla.

3.2 Ajan huomioiminen

Malli voi ottaa huomioon ajan kulumisen liikkeessä. Toisaalta malli voi perustua pelkästään paikka- ja nopeusvektoreihin. Aika voidaan ottaa huomioon myös välillisesti vaihemuuttujan avulla. Vaihe voi riippua suoraan pelkästään ajasta, mutta se voi ottaa huomioon myös paikka- ja nopeusvektorit. Ajan huomioiminen mahdollistaa sellaiset toiminnot kuten ajan keinotekoisen nopeuttamisen tai hidastamisen, mutta myös myös samojen paikka- ja nopeuskoordinaattien läpikäymisen useaan kertaan eri tavoilla.



Kuva 2: Punaisella tähtäimellä osoitettu maalipiste voidaan saavuttaa kahdella erilaisella nivelten asennoilla.

3.3 Stabiilisuus

Usein liikkeelle määritellään niin sanottu maalipiste, johon toteutettavan liikkeen on tarkoitus päättyä. Stabiilisuus tarkoittaa sitä, että mahdollisista häiriöistä huolimatta lopulta päädytään joka tapauksessa maalipisteeseen. Stabiilisuus voi olla lokaalia tai globaalia. Stabiilisuus on selvästikin toivottava ominaisuus mallissa. Vaihtelevan aloituspisteen lisäksi täytyy huomioida häiriöt kesken liikkeen suorittamisen. Robottikäsi voi mahdollisesti siirtyä ulkoisen tai sisäisen häiriön takia paikaltaan tai sen nopeus voi vaihdella tavoitellusta. Pienet häiriöt ovat luonnollisia jo liukulukulaskennan ja fyysisen kokoonpanon epätarkkuuden takia. Teollisuusroboteille onkin määritelty standardissa ISO 9283:1998 toistettavuus, joka tarkoittaa paikan eroavaisuutta toistojen paikan keskiarvosta. Toistettavuus kertoo siis käytännössä, kuinka suuria virheitä robotin liikkeille on odotettavissa.[6]

3.4 Laskennan kompleksisuus

Laskenta voidaan suorittaa kahdella eri tavalla. Robotissa reaaliajassa tapahtuvalle laskennalle kompleksisuudella on väliä. Toisessa tapauksessa laskenta voidaan suorittaa erikseen ennen kuin robotin on aika suorittaa liikkeitä. Tällöin laskennan vaativuudelle ei ole yhtä tiukkoja rajoituksia. Tavallisesti

voidaan ajatella, että laskenta voidaan suorittaa rauhassa, ellei robotin pidä toistaa reaaliajassa jotakin esimerkkiä.

4 Dynaamiset alkeisliikkeet (DMP)

4.1 Malli

Eräs malli liikeratojen toistamiseen on dynaamisten alkeisliikkeiden malli (Dynamical Movement Primitives, DMP). Mallia ovat kehittäneet etenkin Stefan Schaal ja Auke Jan Ijspeert.[5] Malli, jota käsittelen tässä on Sylvain Calinonin versio mallista.[2] Kutsutaan sitä nimellä DMP-C.

DMP-C perustuu siihen, että tavoiteltava liikerata jaetaan ennalta valittavaan määrään attraktoreita. Nämä attraktorit jaetaan tavallisesti tasaisin aikavälein, niin että yksi on heti alussa ja yksi lopussa. Kun aika etenee eri attraktorit vetävät liikettä puoleensa toisen asteen dynamiikan mukaisesti. Näin päästään asteittain alusta loppuun. Lisäämällä attraktoreiden määrää järjestelmä saadaan mielivaltaisen tarkaksi. Mallin formaali määrittely esitellään seuraavasti.[2]

$$\ddot{x} = \sum_{i=1}^{N} h_i(t)[kp_i(\mu_i - x) - kv\dot{x}] \tag{1}$$

Kiihtyvyys kullakin ajanhetkellä kullekin koordinaatille lasketaan kaavan (1) avulla. Käytännössä kaava on painotettu summa harmonisten oskillaattoreiden vaikutuksista. Oskillaattoreiden keskipisteet eli attraktorit ovat μ_i ja ne määrätään esimerkkiaineiston perusteella yksinkertaisimmillaan pienimmän neliösumman menetelmällä.[2]

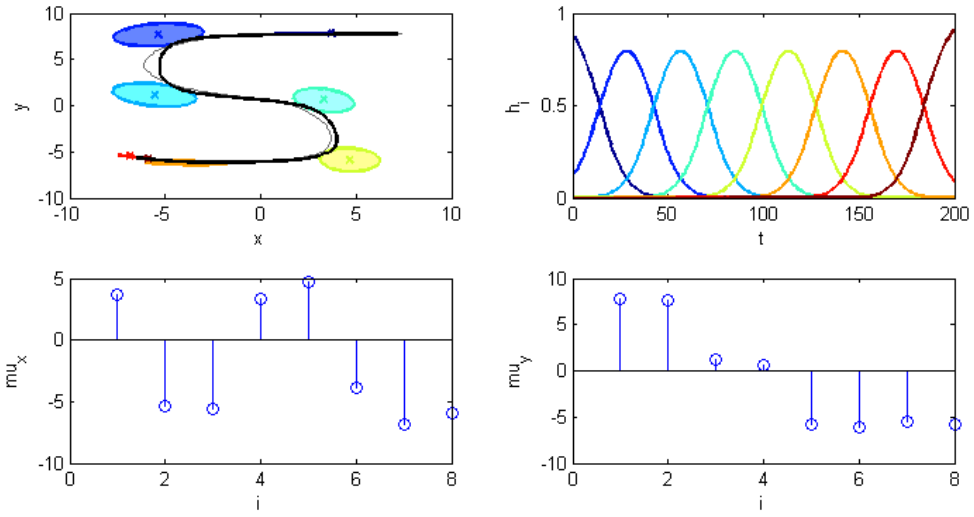
Kertoimet kv ja kp_i ovat kullekin attraktorille ominaiset vakiot, joista kv valitaan siten, että jousi on kriittisesti vaimentunut. kp_i voi vaihdella attraktorista toiseen, jos halutaan ottaa huomioon aineiston varianssi useamman esimerkin tapauksessa. Tässä työssä käsitellään yksinkertaisuuden vuoksi vain tapausta, jossa molemmat kertoimet ovat vakioita.[2]

Painokertoimet $h_i(t)$ normalisoidaan yhteen ja ne määritellään normaalija-

kauman avulla

$$h_i(t) = \frac{N(t; \mu_i, \sigma_i)}{\sum_{i=1}^{N} N(t; \mu_i, \sigma_i)}. \tag{2}$$

Normaalijakaumien odotusarvot on jaettu tasavälein aika-avaruuteen. Varianssiparametrit määritellään käänteisesti attraktorien määrään verrannollisena. Tavallisesti voidaan asettaa esimerkiksi niin, että jakaumat ovat yhden keskihajonnan päässä toisistaan.[2]



Kuva 3: Vasemmalla ylhäällä: esimerkkireitti harmaalla ja toteutettu reitti mustalla sekä attraktorien paikat x-y-koordinaatistossa. Oikealla ylhäällä: attraktorien voimakkuudet ajan funktiona. Alhaalla: attraktorien paikat x- ja y-koordinaatistoissa.

Kuvassa 3 on esitetty esimerkin avulla, kuinka DMP-C käytännössä toimii. Kuvan vasemmassa yläkulmassa on esitetty malliliikerata harmaalla viivalla ja sen lopullinen toteutus mustalla viivalla. Laskennassa on käytetty päätepisteiden lisäksi yhteensä kuutta attraktoria. Liike alkaa liikkeen oikeasta yläkulmasta ja päättyy vasempaan alakulmaan. Attraktorien voimakkuudet muuttuvat ajan funktiona. Kuvan alarivillä on vielä esitetty attraktorien keskipisteet kussakin koordinaatissa. Tämä sisältää saman informaation kuin liikeratakartan soikiomaiset attraktorit.

4.2 DMP-C:n ominaisuudet

4.2.1 Aika

DMP-C tiedostaa ajan. Aika toteutetaan tavallisesti vaiheena. Mallissa on vaiheelle s aikariippuvuus

$$t = \frac{ln(s)}{\alpha}, \tag{3}$$

missä α on jokin vakio. Aikaa voidaan siten hallita tämän vaiheen kautta. Periaatteessa tekoäly voisi tunnistaa tilanteen, jossa vaihe pitää pysäyttää esimerkiksi sen takia, että reitille on tullut este. Vaihe pysäytettäisiin jolloin liikerata pysähtyisi päällä olevaan attraktoriin. Kun vaiheen annetaan edetä, liike voisi taas jatkua normaalisti. Mikäli vaihetta ei pysäytettäisi, järjestelmä pyrkisi jatkamaan liikettä ja esimerkiksi kuvan 3 tapauksessa S-kuvion alempi lenkki saattaisi jäädä väliin, kun vaihe ohittaisi nämä attraktorit robotin ollessa jumissa.

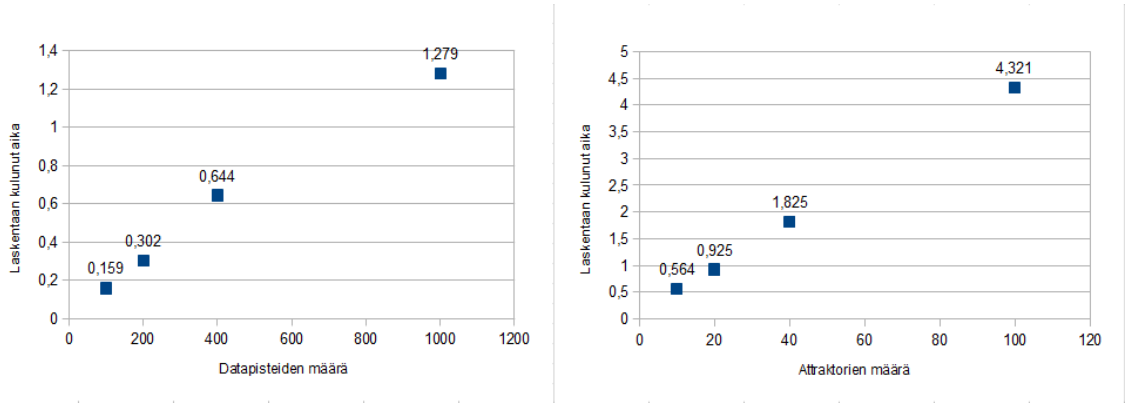
Vaihe voidaan myös muokata sykliseksi.[2] Tämä voidaan toteuttaa esimerkiksi modulaariaritmetiikalla vai syklisen funktion avulla. Syklisellä vaiheella voidaan taas toteuttaa syklisiä liikkeitä kuten rummutusta tai raajojen liikkeitä kävelyssä.

4.2.2 Stabiilisuus

DMP-C on globaalisti stabiili. Tämän näkee siitä, että viimeinen attraktori dominoi aina lopussa samalla kun muut kuihtuvat pois. Täten järjestelmä tuntee kiihtyvyyttä lopussa vain toistettavan liikeradan viimeistä pistettä kohti. Tämä takaa konvergoinnin maalipisteeseen. Mikäli suorituksessa käytetään jonkinlaista heurestiikkaa vaiheen päättelemiseen suoraan ajan kautta määrittelemisen sijasta, voi käydä niin, ettei vaihe etene loppuun saakka. Normaalisti voidaan kuitenkin olettaa vaiheen etenevän loppuun saakka.

4.2.3 Ulottuvuudet

DMP-C:ssä ulottuvuudet ovat riippumattomia. Sillä, käytetäänkö reaalikoordinaatistoa vai nivelkoordinaatistoa, ei ole väliä mallin kannalta. Nivelkoordinaatistossa ulottuvuuksia olisi enemmän, mutta samalla laskenta tarkentuisi,



Kuva 4: Vasemmalla: laskenta-aika datapisteiden funktiona. Oikealla: laskenta-aika attraktorien funktiona. Molemmat ovat käytännössä lineaarisia riippuvuuksia käytetyllä alueella.

jos muut tekijät pysyisivät samoina. Käytännössä kummankaan koordinaatiston valinnan ei pitäisi aiheuttaa hankaluuksia.

4.2.4 Laskennallinen kompleksisuus

DMP-C:n laskennallinen kompleksisuus on suunnilleen $O(nmp)$, missä n on datapisteiden, m attraktorien ja p ulottuvuuksien määrä. Tämä tulee siitä, kun lasketaan jokaiselle toiston datapisteelle kiihtyvyys ja kiihtyvyyden laskennassa tarvitaan jokaisen attraktorin painokerrointa. Lisäksi tämä kaikki käydään läpi jokaiselle ulottuvuudelle. Kuvassa 4 on esitetty laskenta-ajan riippuvuus datapisteiden määrästä ja attraktorien määrästä riippuvana kokeellisesti.

4.2.5 Varianssi

DMP-C ottaa oletuksena huomioon usean esimerkkiliikeradan vaihtelut eri kohdissa liikerataa. Tätä tietoa voidaan käyttää hyväksi, jos liikeradan toistoon halutaan lisätä potentiaaliin perustuva väistöjärjestelmä. Toinen järjestelmä voi havainnoida uhkia liikkelle ja muodostaa jonkinlaisen luotaan työntävän potentiaalin näiden ympärille. Mitä enemmän vaihtelua tietyssä osassa esimerkkiliikerataa on ollut, sitä helpommin uhkalle annetaan periksi liikeradan noudattamisen sijasta.

4.3 Vertailu alkuperäiseen DMP:hen

DMP-C eroaa Stefan Schaalin ja Auke Jan Ijspeertin DMP:stä (DMP-S) osittain. DMP-S on määritelty myös toisen asteen jousisysteeminä[5]

$$\tau\ddot{y} = \alpha_z(\beta_z(g - y) - \dot{y}) + f. \tag{4}$$

Tässä g on maalipiste ja y paikkakoordinaatti. α_z ja β_z ovat jousisysteemin vakioita, jotka valitaan niin, että systeemi on kriittinen. τ on aikavakio, joka voidaan käytännössä piilottaa muihin vakioihin. Funktiota $f(x)$ kutsutaan pakotustermiksi ja se määritellään[5]

$$f(x) = \frac{\sum_{i=1}^{N} \Psi_i(x)\omega_i}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0), \tag{5}$$

N:llä gaussisella kantafunktiolla

$$\Psi_i(x) = exp(-\frac{1}{2\sigma_i^2}(x - c_i)^2). \tag{6}$$

Kaavassa (6) σ_i ja c_i ovat vakioita, jotka määrittelevät kantafunktioiden leveyden ja paikan. Kaavassa (5) y_0 on alkuperäinen tila hetkellä $t = 0$.[5]

DMP-S eroaa DMP-C:stä etenkin maalipisteen huomioimisen suhteen. DMP-S:ssä maalipiste löytyy pakotustermistä. Termiä kerrotaan maalinpisteen ja aloituspisteen erotuksella. Siten aloituspiste itsessään vaikuttaa rataan koko liikeradan ajan.[5] Tämä ominaisuus aiheuttaa eräänlaisen skaalauksen mallissa. Esimerkiksi yksinkertaista siniaaltoa toistettaessa aloituspisteen siirtäminen pisteestä $y = 1$ pisteeseen $y = 2$ kaksinkertaistaisi suoraan toiston amplitudin. Tässä muodossa malli säilyttäisi siten liikeradan muodon paremmin kuin paikkakoordinaatit.

5 Dynaamisten järjestelmien stabiili estimaattori (SEDS)

5.1 Malli

Dynaamisten järjestelmien stabiili estimaattori (Stable Estimator of Dynamical Systems, SEDS)[8] on metodi, jossa liike mallinnetaan epälineaarisena aikainvarianttina dynaamisena systeeminä. Mallin esittelivät ensimmäisenä S. Mohammad Khansari-Zadeh ja Aude Billard.

Malli määritellään tilamuuttujan ξ avulla differentiaaliyhtälönä

$$\dot{\xi} = f(\xi, \theta) = \sum_{k=1}^{K} \frac{P(k)P(\xi|k)}{\sum_{i=1}^{K} P(i)P(\xi|i)} (\Sigma_{\dot{\xi}\xi}^{k} (\Sigma_{\xi}^{k})^{-1} \xi + \mu_{\dot{\xi}}^{k} - \Sigma_{\dot{\xi}\xi}^{k} (\Sigma_{\xi}^{k})^{-1} \mu_{\xi}^{k}). \quad (7)$$

Tämä voidaan merkitä muotoon

$$\dot{\xi} = f(\xi, \theta) = \sum_{k=1}^{K} h^{k}(\xi)(A^{k}\xi + b^{k}). \quad (8)$$

Tässä θ:lla on merkitty opittavia parametreja

$$\theta = \{\pi^{1}..\pi^{K}, \mu^{1}..\mu^{K}, \Sigma^{1}..\Sigma^{K}\} \quad (9)$$

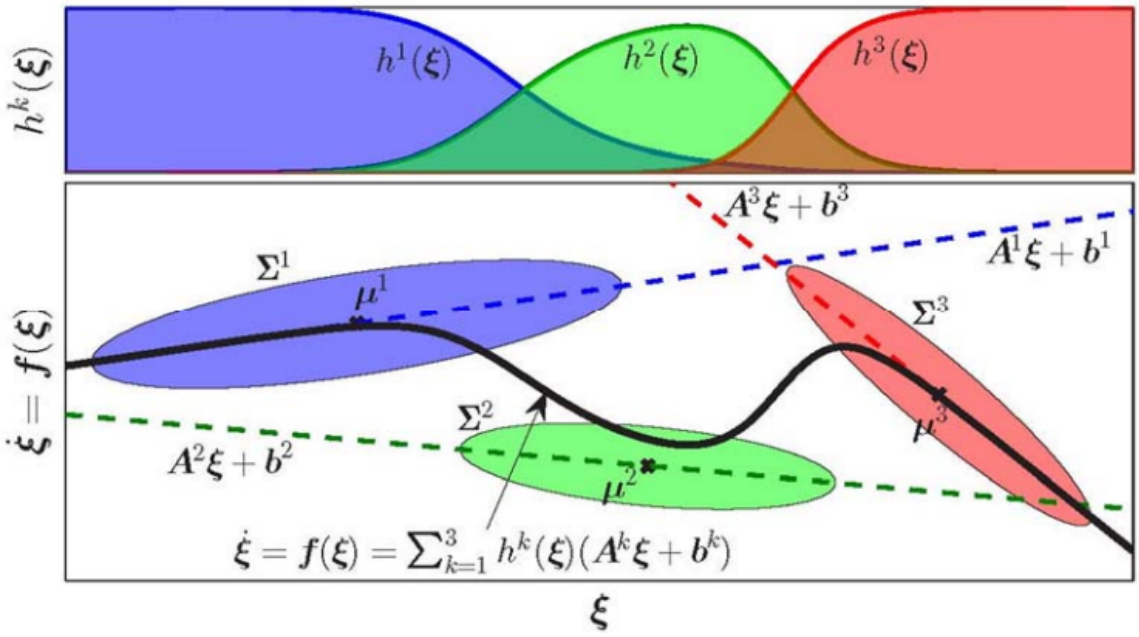
Kuvassa 5 on esitetty optimoitavien parametrien konkreettista merkitystä. h^{k} kertoo käytännössä, miten voimakkaasti kukin osa on kullakin hetkellä voimassa. $(A^{k}\xi + b^{k})$ edustaa kukin gaussista funktiota.

Lyapunovin stabiilisuusteoreeman avulla saadaan systeemiin kaksi rajoitusehtoa

$$(a)\ \mu_{\dot{\xi}}^{k} = \Sigma_{\dot{\xi}\xi}^{k} (\Sigma_{\xi}^{k})^{-1} (\mu_{\xi}^{k} - \xi^{*}) \quad (10)$$

ja

$$(b)\ \Sigma_{\dot{\xi}\xi}^{k} (\Sigma_{\xi}^{k})^{-1} + (\Sigma_{\xi}^{k})^{-1} (\Sigma_{\dot{\xi}\xi}^{k})^{T} \prec 0. \quad (11)$$



Kuva 5: Ylhäällä: havaintokuva painokerrointen h^k havainnollistamiseksi. Alhaalla: havaintokuva muiden osien havainnollistamiseksi.

Nämä ehdot takaavat sen, että tehtävän ratkaisu on globaalisti stabiili.[8]

Tätä systeemiä voidaan optimoida esimerkiksi pienimmän neliösumman menetelmällä (SEDS-MSE) tai suurimman uskottavuuden menetelmällä (SEDS-Likelihood). Malli mahdollistaa suoraan mielivaltaisen monen esimerkkiliikeradan käytön. Mikäli mallia optimoidaan suurimman uskottavuuden menetelmällä, optimointitehtäväksi saadaan

$$min\ J(\theta) = -\frac{1}{\tau}\sum_{n=1}^{N}\sum_{t=0}^{T^n} logP([\xi^{t,n}; \dot{\xi^{t,n}}]|\theta) \tag{12}$$

rajoitusehdoilla

$$\begin{cases} \mu_{\dot{\xi}}^k = \Sigma_{\dot{\xi}\xi}^k(\Sigma_{\xi}^k)^{-1}(\mu_{\xi}^k - \xi^*) \\ \Sigma_{\dot{\xi}\xi}^k(\Sigma_{\xi}^k)^{-1} + (\Sigma_{\xi}^k)^{-1}(\Sigma_{\dot{\xi}\xi}^k)^T \prec 0 \\ \Sigma^k \succ 0 \qquad\qquad \forall k \in 1..K \\ 0 < \pi^k \leq 1 \\ \sum_{k=1}^{K} \pi^k = 1 \end{cases} \tag{13}$$

5.2 SEDSin ominaisuudet

5.2.1 Aika ja ulottuvuudet

SEDS tuottaa ratkaisuna aikainvariantin tuloksen. Tämän lisäksi ratkaisu on paikka-avaruudessa. Tämä johtaa siihen, että ratkaisuna saatava liikerata ei voi käydä samassa pisteessä kahdesti. Siten SEDS ei voi tuottaa esimerkiksi kahdeksikon muotoista rataa. Tämä on mahdollista ohittaa muuttamalla systeemi toisen kertaluvun differentiaaliyhtälöksi, joka voidaan edelleen muuttaa ensimmäisen kertaluvun differentiaaliyhtälöryhmäksi.[8] Toinen vaihtoehto olisi lisätä systeemin vaihemuuttuja.[8]

Menetelmä on rakenteeltaan moniulotteinen. Yhdessä ulottuvuudessa SEDS ei ole mielekäs vaihtoehto, sillä se sallii liikkeen vain yhteen suuntaan ja vain monotonisten funktioiden toistaminen on mahdollista. Siksi SEDSiä käyttäessä kaikki ulottuvuudet kannattaakin käsitellä samalla, jotta tästä rajoitteesta pääsee eroon.

5.2.2 Optimointi

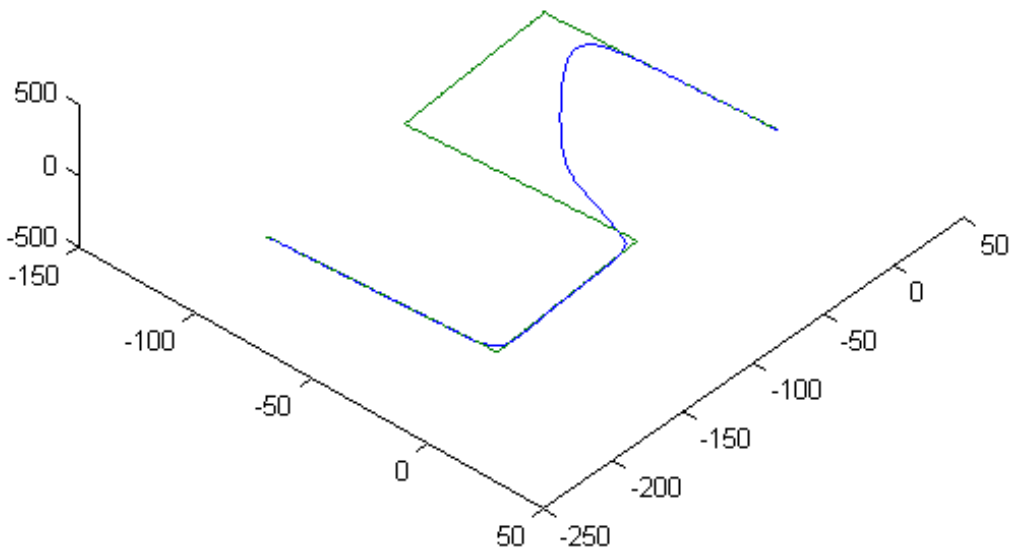
SEDS ei itsessään ota kantaa siihen, millä algoritmilla itse optimointi toteutetaan. Optimointialgoritmin valinta voi vaikuttaa lopputulokseen ja eri algoritmeilla on hyvät ja huonot puolensa. Billard ja Khansari-Zadeh käyttivät itse toistetun neliöllisen ohjelmoinnin menetelmää julkaisussaan.[8]

Optimoidessa täytyy ensin valita haluttu määrä gaussisia funktioita. Tekijät käyttivät bayesilaista informaatiokriteeriä tätä varten (Bayesian Information Criterion, BIC).[8] Tällä tavoin myös tämä vaihe voidaan automatisoida.

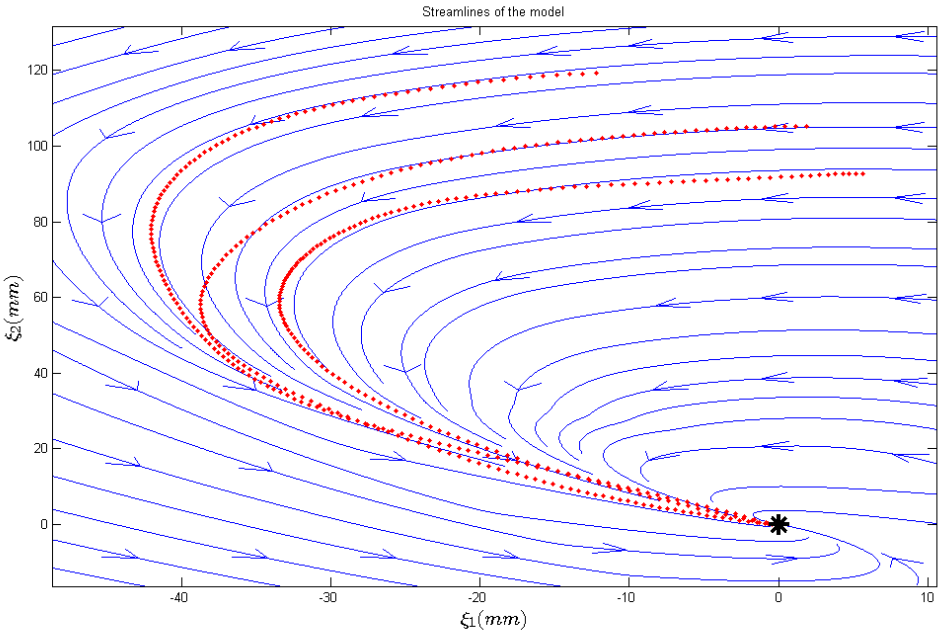
Käytettäessä SEDSin suurimman uskottavuuden menetelmää estimoitavien parametrien määrä on $K(1 + 3d + 1d^2)$, missä K on gaussisten funktioiden määrä ja priorit π^k on kokoa 1, odotusarvot μ^k kokoa 2d ja kovarianssit Σ^k kokoa d(2d+1). Laskemalla odotusarvot rajoitusehdoista saatavilla lausekkeilla muiden parametrien arvoista estimoitavien parametrien määräksi saadaan vähennettyä $K(1 + 2d(d + 1))$.

SEDSin pienimmän neliösumman menetelmässä parametrien määrä saadaan vielä alhaisemmaksi arvoon $K(1 + \frac{3}{2}d(d + 1))$.[8]

Koska optimoitavana on yleinen epälineaarinen optimointitehtävä, ei ole takeita siitä että löydettävä ratkaisu on globaali optimi. Ratkaisumenetelmät ovat tavallisesti hyvin herkkiä parametrien alkuarvoille. Käytännön seuraus



Kuva 6: Optimointitehtävän ratkaisu juuttuu usein paikalliseen optimiin. Tässä stabiilisuusehdot aiheuttivat ongelmia globaalin optimin löytämisessä. Vihreällä on esitetty tavoiteltava liikerata ja sinisellä mallin antama liikerata.



Kuva 7: Kaikista mahdollisista aloituspisteistä päädytään maaliin.

tästä on se, että aina SEDS ei kykene löytämään ratkaisua, joka seuraisi annettua liikerataa halutun tarkasti. Kuvassa 6 on esitetty yksi ratkaisu, kun annettu liikerata on ollut kulmikas. Alkuvaiheessa liikerataa ratkaisu noudattaa haluttua reittiä, mutta loppupäässä ratkaisu oikaisee maalipisteeseen kulkematta kunnolla rataa läpi.

5.2.3 Stabiilisuus

SEDS eroaa muista gaussisista sekamalleista (Gaussian Mixture Models) juuri siinä, että se takaa globaalin stabiilisuuden. Muista menetelmistä Gaussian Process Regression, Locally Weighted Projection Regression ja Gaussian Mixture Regression eivät takaa edes lokaalia stabiilisuutta. Binary Merging takaa ainakin lokaalin asymptoottisen stabiilisuuden. Kuvassa 7 on esitetty mallilla saatu tulos, kun esimerkkinä on kolme C-kirjaimen muotoista liikerataa maalipisteeseen. Nähdään, että vaikka aloituspiste olisi mielivaltainen, päädytään silti aina lopulta maalipisteeseen.

6 Rekurrentti neuroverkko parametriharhalla (RNNPB)

6.1 Malli

Rekurrentti neuroverkko parametriharhalla (Recurrent Neural Network with Parametric Bias, RNNPB) on neuroverkkoihin perustuva lähtökohta liikeratojen mallintamiseen. RNNPB on arkkitehtuuriltaan samanlainen kuin tavallinen Jordan-tyyppinen toistuva neuroverkko. RNNPB:ssä on lisätty vain parametriharhasolmuja syötekerrokseen.

Kuvassa 8 on esitetty kuvaus menetelmän arkkitehtuurista (a) oppimisvaiheesta ja (b) vuorovaikutusvaiheessa. Oppimisvaiheessa RNNPB harjoitetaan käytännössä harjoitusaineistolla eli esimerkki liikeradoilla. Vuorovaikutusvaiheessa menetelmä tuottaa ulostuloja sisääntulojen perusteella. Tavallisille sisään- ja ulostulosolmuille suoritetaan kahdenlaisia operaatioita: suljetun silmukan ja avoimen silmukan operaatioita. Lisäksi arkkitehtuuriin kuuluu kontekstisolmuja c_i sekä sisääntulo että ulostulokerroksissa. Menetelmän nimi tulee parametriharhasolmuista p_t, joita on sisääntulokerroksessa. Näitä solmuja voi manipuloida monipuolisten käytösten opettamiseksi ja tuottamiseksi.[7]

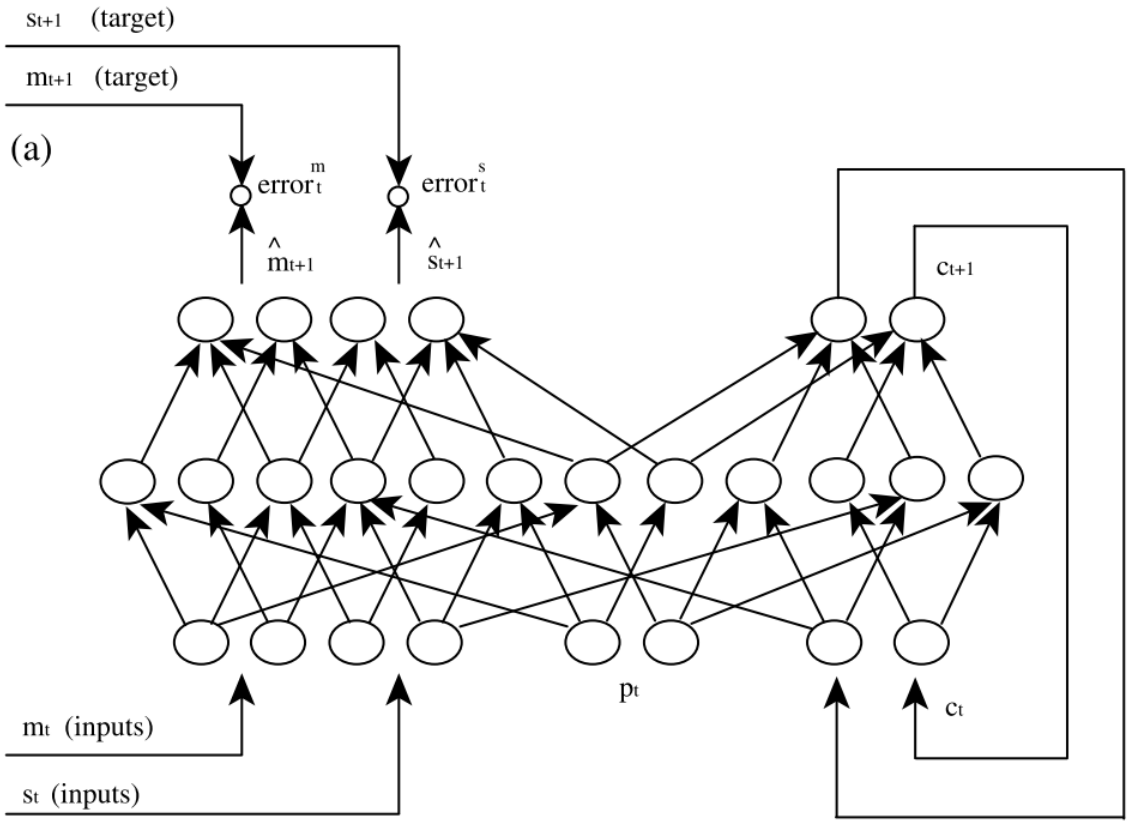
Harjoitusaineiston ominaisuudet hankitaan backpropagation through time -algoritmilla (BPTT). Samalla jokaisen yksittäisen aikasarjan tietyt ominaisuudet koodataan harhaparametrien arvoihin. Oppimis- ja tunnistusprosesseissa harhaparametriarvot lasketaan iteratiivisesti käyttämällä virhettä ennustetun ja tavoitellun sarjan välillä.[7]

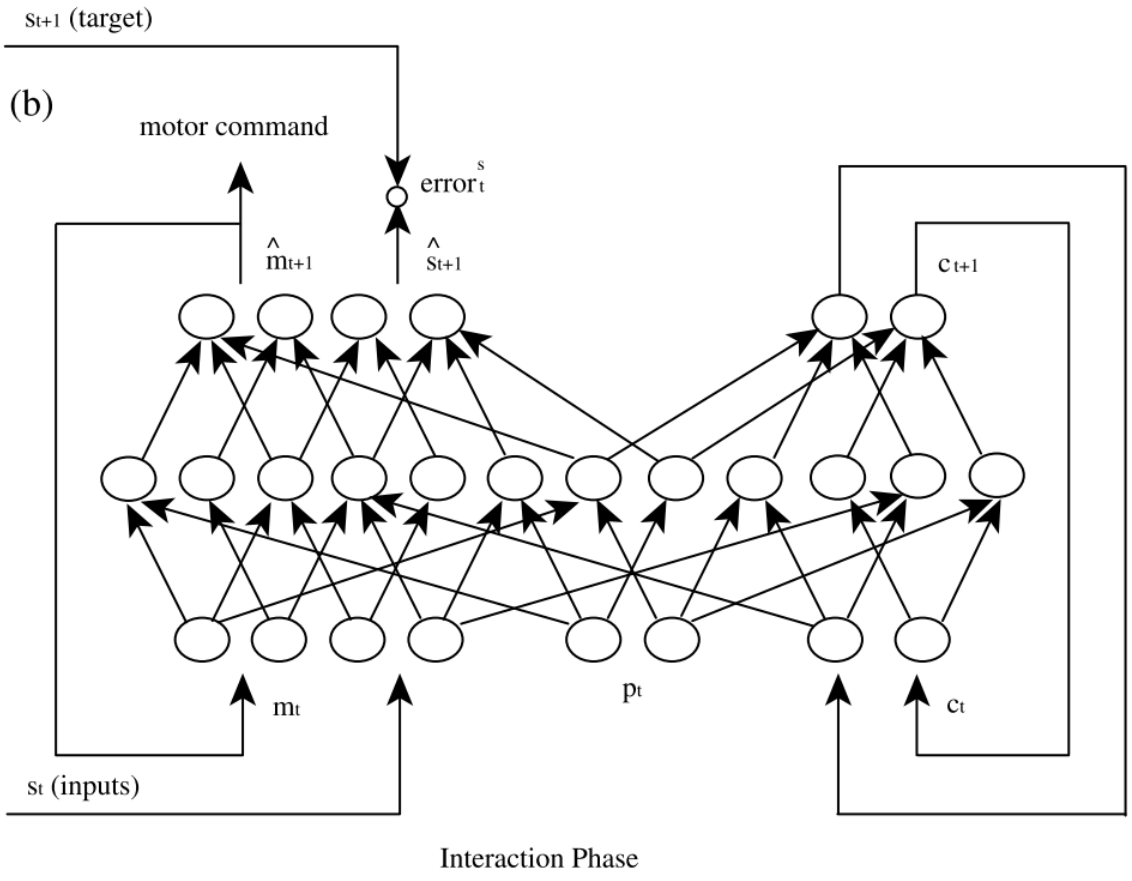
6.2 RNNPB:n ominaisuudet

6.2.1 Useat liikeradat

RNNPB mahdollistaa siirtymisen usean eri liikkeen välillä. Tämä luo sekä mahdollisuuksia että ongelmia. Toisaalta menetelmää voidaan ulkoisesti ohjata liikeradalta toiselle. Toisaalta taas malli voi ajautua ongelmiin liian monen erilaisen liikkeen kanssa. Menetelmä voi harhautua liikkeeltä toiselle tahattomasti. Se, miten liikkeestä toiseen voi vaihtaa, riippuu liikkeen pysyvyydestä. Myös sillä, missä vaiheessa liikettä ollaan, on väliä.[7]

Muita useiden liikkeiden oppimiseen käytettäviä metodeja ovat MOSAIC (Wolpert ja Kawato, 1998) ja





Kuva 8: Kuvaus RNNPB:n (a) oppimis- ja (b) vuorovaikutusvaiheesta.

6.2.2 Yleistäminen

RNNPB on neuroverkkomalli, minkä takia sille ei ole johdettu tiettyjä teoreettisia ominaisuuksia. Simulaatioilla on voitu osoittaa, että RNNPB on melko stabiili häiriöitä vastaan liikkeen valinnassa. Malli kykenee yleistämään siniaaltoisen liikkeen taajuuden, mutta amplitudin yleistämisessä teho on rajattu.[4]

6.2.3 Muut neuroverkot

Tavallisesti viiveneuroverkkoratkaisut (Time Delay Neuro Networks) vaativat hyvin suuren määrän neuroneita ja oppimisaikaa, sillä ne on tarkoitettu säilyttämään kaikki aikasarja-aineisto syötekerroksessa. RNNPB käyttää itsejärjestäytyvää kontekstuaalista informaatiota kontekstikerroksessa, joten se pystyi selviytymään eräässä testissä objektin lyömisestä vain 42 neuronilla.[10]

Tavallisesti neuroverkkoratkaisuilla on ongelmia pitkän aikavälin riippuvuuksia vaativissa mallinnuksissa, koska virhesignaaleja ei pystytä tehokkaasti välittämään BPTT-algoritmilla mahdollisten epälineaarisuuksien takia. Eräs mahdollinen ratkaisu ongelmaan on Long Short-Term Memory (LSTM) -metodi. Muita samantyyppisiä vaihtoehtoja ovat kaikutilaverkot (Echo state networks) ja nestetilakoneet (liquid state machines). On osoitettu, että kaikutilaverkko pystyy onnistuneesti oppimaan Mackeyn-Glassin kaoottisen aikasarjan, mikä on tunnettu perustesti aikasarjan ennustamisessa.[9]

On esitetty malleja, joissa yhdistetään useita eri kerroksia neuroverkkomalleja yhteen. Tavallisesti ongelmana on skaalautuessa ilmeentyvät stabiilisuusongelmat. Niin sanottu RNN experts -malli pyrkii välttämään tämän. Yksinkertaistettuna mallissa jokaisellla tasolla eri neuroverkot kilpailevat toisiaan vastaan ja korkeimmilla tasoilla aikavakiot ovat pienemmät. Alemmat tasot oppivat yksittäisiä liikkeitä, kun korkeammat tasot voivat oppia näiden primitiivisten osien abstraktioita. Myös muita samankaltaisia malleja on. Neuroverkkoratkaisun skaalautumisominaisuudet määrittelevät pitkälti mallin kelpoisuuden skaalautuviin ongelmiin. On toivottavaa, että oppimisprosessi on stabiili suurellakin määrällä moduuleita.[9]

7 Vaihtoehtoisia malleja

Näyttämällä opettamisen mallit voidaan jakaa kahteen lajiin: insinöörilähtöisiin ja biologialähtöisiin malleihin. Insinöörilähtöisissä tavoissa keskitytään kehittämään algoritmeja, jotka ovat yleisiä esityksiä taidoista. Nämä voidaan edelleen jakaa symboliseen koodaukseen ja liikeradan koodaukseen. Biologiset mallit hakevat inspiraationsa eläinten tavasta imitoida asioita.[1]

Tässä työssä esitetyistä malleista DMP-C ja SEDS voidaan laskea insinöörilähtöisiksi malleiksi. RNNPB on neuroverkkomallina sen sijaan eräässä mielessä biologiseen perustaan nojaava.[1]

Alexander Skoglund, Boyko Iliev ja Rainer Palm ovat käyttäneet sumeaa mallintamista sekä liikkeiden havaitsemiseen että niiden toteuttamiseen. Tämä onnistui siitä huolimatta, että mallina olevan ihmisen ja liikkeen toistavan robotin anatomiassa oli eroavaisuuksia.[11]

Monet mallit yhdistävät nykyään peilisolut ja robottien ohjauksen. Mallit auttavat sekä selittämään peilisoluja eläimissä että käyttämään tätä tietoa ohjauksessa. Myös evoluutiota on yhdistetty tähän.[1]

Useat eri tutkijat ovat käyttäneet Markovin piilomalleja tilastopohjaisen oppimisen perusteena. Markovin piilomallit ovat sopivan robusteja aika- ja paikkavarianteille signaaleille.[1]

8 Vertailu

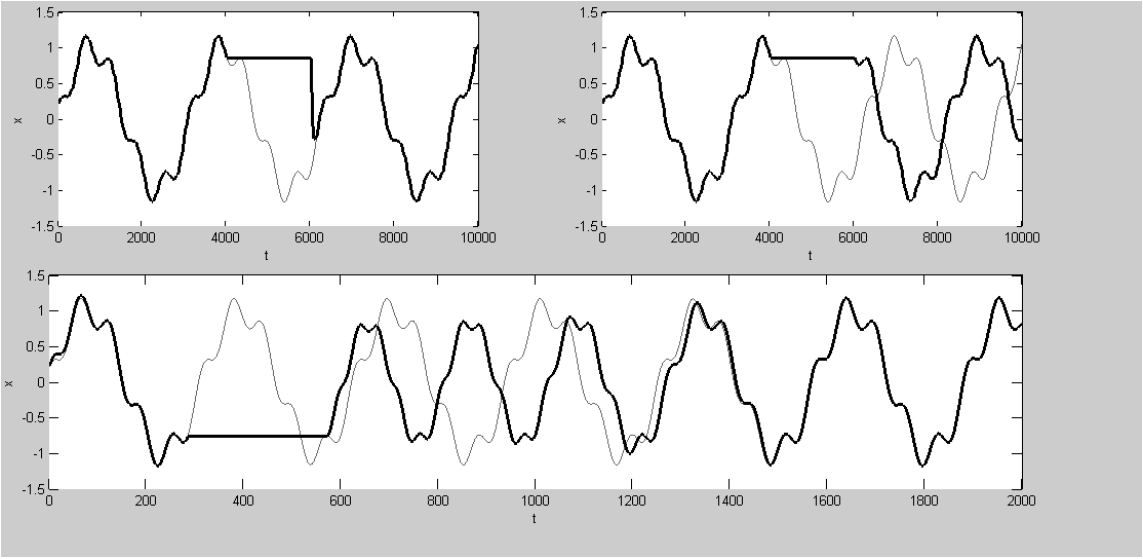
Esitellyt kolme mallia ovat hyvin erilaiset. DMP-C perustuu toisen asteen dynamiikkaan, SEDS epälineaariseen optimointiin ja RNNPB neuroverkkoihin. Tästä syystä myös mallien ominaisuudet eroavat paljon.

Malleista DMP-C ja SEDS ovat globaalisti stabiileja. RNNPB sen sijaan ei pysty takaamaan stabiilisuutta. Stabiilisuus ei välttämättä ole tarpeellista, jos tarkoituksena on toistaa vain liikeradan yleisiä ominaisuuksia, kuten muotoja. Esimerkiksi allekirjoituksen toistamisessa loppupisteellä ei ole niinkään väliä: merkitseviä ovat kirjainten kaaret. Sen sijaan vaikkapa objektin siirtämisessä paikasta toiseen loppupaikka on merkittävässä asemassa ja tällaisessa tehtävässä on hyvä käyttää stabiilia mallia.

Malleista vain DMP-C on aikariippuvainen. Aikariippuvuus mahdollistaa monia toimintatapoja virhetilanteissa. Kuvassa 9 on esitetty kolme vaihtoehtoista tapaa käsitellä hetkittäinen jumiutuminen DMP-C:n avulla. Ylhäällä esi-

tetyt tavat ovat tavallisimpia. Vasemmalla ylhäällä mallin sisäinen kello jatkaa etenemistä jumiutumisen aikanakin, jolloin liikerata ohittaa tällä välillä olleet liikeradan osat. Oikealla ylhäällä mallin sisäinen kello pysähtyy. Tällöin kaikki liikeradan osat käydään tavallisesti läpi, mutta liike on ajallisesti jäljessä esimerkkiliikettä. Alhaalla käytetään kahta sisäistä kelloa. Toinen kelloista pysäytetään ja toista ei. Päästäessä jatkamaan taas liikettä oikea aika otetaan asteittain kiinni, jolloin kaikki halutut liikkeen osat käydään läpi ja lopulta liike on myös ajallisesti oikeassa. Tämä vaihtoehto vaatii muun muassa virhetilanteen tunnistamista sekä valintaa siitä, kuinka nopeasti aikaero halutaan kuroa kiinni, joten tilanne monimutkaistuu.

Malleista DMP-C on laskennallisesti kevyin. Se ei tarvitse raskaita epälineaaristen tehtävien ratkaisumenetelmiä toisin kuin SEDS. Neuroverkkoratkaisuna myös RNNPB vaatii melko raskaita laskutoimituksia. Jos laskenta toteutetaan niin sanotusti offline-laskentana muualla kuin robotissa itsessään, tällä ei välttämättä ole väliä. Kaikissa metodeissa laskentaa voidaan aina keventää vähentämällä aineistoa. Datapisteistä voidaan jättää esimerkiksi merkittävä osa käyttämättä tilanteesta riippuen.



Kuva 9: Vasemmalla ylhäällä: edetään kuin aika olisi jatkanut etenemistään normaalisti. Oikealla ylhäällä: pysäytetään aika. Alhaalla: otetaan rytmi paloittain kiinni.

9 Yhteenveto

Työssä esiteltiin ensiksi yleisiä periaatteita liikeratojen toistamisesta robotin avulla. Robottien asettamia rajoituksia ja merkitystä malleille käytiin läpi.

Erilaisista liikeratojen toistamisen malleista esiteltiin kolme: DMP-C, SEDS ja RNNPB. Näiden ominaisuudet esiteltiin ja niiden merkitystä pohdittiin. Myös vaihtoehtoisia malleja käytiin lyhyesti läpi. Valituista kolmesta mallista DMP-C on luultavasti yksinkertaisin ja mielekkäin toteuttaa ensimmäisenä kokeiluna. Malli toimii mielivaltaisilla liikeradoilla ja mahdollistaa myös erilaiset laajennukset ajan käsittelyn suhteen.

Malleja ajatellessa on hyvä aina kokeilla niitä myös käytännössä. Tässä käytäntö tarkoittaisi simulaatioiden lisäksi myös implementoimista robottiin ja käytännön kokeita liikeratojen toistamisessa. Tässä työssä rajoituttiin vain mallien tarkasteluun matemaattisina olioina. Malleja on kokeiltu myös empiirisesti niiden kehittäjien toimesta.

Viitteet

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. Handbook of Robotics, . chapter 59, 2008, 2008.

[2] S. Calinon, I. Sardellitti, and D. G. Caldwell. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 249–254, Taipei, Taiwan, October 2010.

[3] John J. Craig. Introduction to robotics: Mechanics and control, 2005.

[4] Raymond H. Cuijpers, Floran Stuijt, and Ida G. Sprinkhuizen-Kuyper. Generalisation of action sequences in rnnpb networks with mirror properties. In *ESANN*, 2009.

[5] A. Ijspeert, J. Nakanishi, P Pastor, H. Hoffmann, and S. Schaal. Dynamical movement primitives: Learning attractor models formotor behaviors. *Neural Computation*, (25):328–373, 2013.

[6] Manipulating industrial robots - Performance criteria and related test methods, ISO 9283:1998, 1998.

[7] Masato Ito, Kuniaki Noda, Yukiko Hoshino, and Jun Tani. Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model, 2006.

[8] S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transaction on Robotics*, (27):943–957, 2011.

[9] J. Namikawa and J. Tani. A model for learning to segment temporal sequences, utilizing a mixture of RNN experts together with adaptive variance. *ArXiv e-prints*, June 2007.

[10] Tetsuya Ogata, Hayato Ohba, Jun Tani, Kazunori Komatani, and Hiroshi G. Okuno. Extracting multi-modal dynamics of objects using rnnpb, 2005.

[11] Alexander Skoglund, Boyko Iliev, and Rainer Palm. Programming-by-demonstration of reaching motions-a next-state-planner approach. *Robot. Auton. Syst.*, 58(5):607–621, May 2010.

[12] Wikipedia. Programming by demonstration — wikipedia, the free encyclopedia, 2013. [Online; Luettu 25.9.2013].

[13] Wikipedia. Robot kinematics — wikipedia, the free encyclopedia, 2013. [Online; Luettu 25.9.2013].