Univerza v Ljubljani
Fakulteta za Računalništvo in Informatiko
Fakulteta za Matematiko in Fiziko

Luka Stopar

Vodenje robota preko senzorjev pametnega telefona

Diplomsko delo

na interdisciplinarnem univerzitetnem študiju
računalništva in matematike

Mentor: akad. prof. dr. Ivan Bratko

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00052/2013
Datum: 08.09.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko ter Fakulteta za matematiko in fiziko izdaja naslednjo nalogo:

Kandidat: **LUKA STOPAR**

Naslov: **VODENJE ROBOTA PREKO SENZORJEV PAMETNEGA TELEFONA**
CONTROLLING A ROBOT USING A SMART PHONE

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Naloga je razviti sistem za vodenje robota s posnemanjem človekovega gibanja, zajetega s senzorji pametnega telefona. Vodenje robota naj poteka tako, da človek z roko izvede gib s telefonom v roki, gib naj se zajame s senzorji telefona, izbrani humanoidni robot pa naj izvede gib čim bolj podoben demonstratorjevemu gibu. Pri izvedbi uporabite robot NAO firme Aldebaran ter naslednje senzorje pametnega telefona: kamera, žiroskop, kompas, pospeškometer. Ocenite uspešnost razvitega sistema in izvedite tudi primerjavo z že obsoječimi sistemi. Pri tem obravnavajte dva vidika uspešnosti: (1) uspešnost samega zajemanja položaja in orientacije telefona, ter (2) uspešnost aplikacije z vodenjem robotove roke.

Mentor:
akad. prof. dr. Ivan Bratko

Dekan Fakultete za računalništvo in informatiko:
prof. dr. Nikolaj Zimic

Dekan Fakultete za matematiko in fiziko:
akad. prof. dr. Franc Forstnerič

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Luka Stopar, z vpisno številko **63070454**, sem avtor diplomskega dela z naslovom:

Vodenje robota preko senzorjev pametnega telefona.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom akad. prof. dr. Ivana Bratka,
- so elektronska oblika diplomskega dela, naslov, povzetek ter ključne besede identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

Ljubljana, 17. september 2013 Podpis avtorja:

Zahvaljujem se mentorju akad. prof. dr. Ivanu Bratku ter Aljažu Košmerlju za vodenje ter nasvete pri izdelavi diplomske naloge.

Zahvala gre tudi sodelavcem Janu, Blažu ter Primožu za vse njihove dobre nasvete in pomoč.

Nazadnje gre zahvala še celotni družini, ki me je podpirala v času šolanja in ni nikoli obupala nad mano. Posebna zahvala gre staršem.

Kazalo

Akronimi in Okrajšave

VM	Navidezni stroj (Virtual Machine)
JVM	Javanski navidezni stroj (Java Virtual Machine)
SDK	Programski razvojni paket (Software Development Kit)
SVD	Singularni razcep (Singular Value Decomposition)
API	Programski vmesnik (Application Programming Interface)

Povzetek

V tej diplomski nalogi smo razvili sistem zajemanja ter repliciranja gibov človeške roke. Zajem gibanja smo implementirali na pametnem telefonu, replikacijo pa na humanoidnem robotu.

Za zajem gibanja smo uporabili pametni telefon, ki vsebuje programski vmesnik (API) za delo s senzorji, kot so: pospeškometer, žiroskop, digitalni kompas ter video kamera. Zajem gibanja smo implementirali v programskem jeziku Java, pri tem smo razvili API za delo z digitalnimi signali ter linearno algebro. Konceptualno smo ga razdelili na dva dela: računanje orientacije ter računanje pozicije.

Za določanje orientacije smo uporabili dva pristopa. Prvi pristop izkorišča meritve žiroskopa. V vsakem časovnem koraku izračuna os in kot vrtenja ter posodobi trenutno orientacijo. Drugi pristop temelji na računanju referenčnih okvirov, kjer izračunamo okvir telesa ter z reševanjem sistema enačb izračunamo orientacijo. Na koncu smo oba pristopa združili, kar nam omogoči potrebno odzivnost ter eliminira zamik žiroskopa.

Pri računanju pozicije smo raziskali dva pristopa. Prvi pristop temelji na računanju pozicije iz meritev pospeškometra. Izračunali smo linearni pospešek v navigacijskem okviru, nato z dvokratnim numeričnim integriranjem izračunali pot od začetne pozicije. Pristop se je izkazal za neuspešnega. Drugi pristop temelji na računanju pozicije s pomočjo računalniškega vida.

Sledili smo štirim vizualnim značilkam vnaprej poznanega vzorca ter tako uspeli izračunati odmik od začetne pozicije, katera se določi ob inicializaciji sistema.

Za replikacijo gibanja smo uporabili humanoidnega robota NAO francoskega podjetja Aldebaran Robotics. Uporabili smo metodo diferenčne kinematike, ki temelji na izračunu psevdoinverza Jacobijeve matrike. Zato, da je sistem robusten na kinematične singularnosti, smo implementirali izračun psevdoinverza z metodo dušenih najmanjših kvadratov, ki s pomočjo singularnih vrednosti Jacobijeve matrike definira regije singularnosti v prostoru sklepov. V teh regijah nato uporabi faktor dušenja, ki je proporcionalen bližini kinematične singularnosti.

Ključne besede: robotika, kinematika, inverzna kinematika, računalniški vid, referenčni okvir, linearni pospešek.

Abstract

In this work, we implemented a motion capture and replication system for human arm movement. Using sensors available on a smartphone, we implemented motion capture and replicated the motion using a NAO humanoid robot developed by the French robotics company Aldebaran Robotics.

The motion capture was implemented using the Java Application Programming Interface (API) provided by the Android system. The sensors include a gyroscope, an accelerometer, a magnetometer and finally a video camera. Conceptually, we divided the motion capture into two parts: determining the hand's orientation and determining its location. For the implementation we developed a specialized API containing methods for linear algebra, quaternions and digital signal processing.

To determine the orientation of the hand, we implemented two approaches: The first approach uses gyroscope readings to compute the rotation axis and angle with which it updates the orientation. The second approach uses accelerometer and magnetometer readings to construct a reference frame. It then solves a system of equations to obtain the current orientation. We combined the two methods to eliminate drift caused by the first approach and ensure responsiveness.

To determine the hand's location we also tried two approaches. The first approach is based on double integration of the accelerometer readings to ob-

tain the location. This approach proved to be unsuccessful, leading to the second approach using computer vision to determine the current location. It tracks four visual features of a pre-defined pattern placed before the device. By knowing the pattern's dimensions, we are able to compute the location.

For motion replication, we used the method of differential kinematics. The method computes a pseudo-inverse of the Jacobian matrix with which it updates the current joint configuration. To make the method robust to kinematic singularities we computed a damped least-squares pseudo-inverse of the Jacobian, which uses singular values to define singularity regions in the joint space. It then uses a damping factor in these regions which is proportional to the proximity of the kinematic singularity.

Keywords: robotics, kinematics, inverse kinematics, computer vision, reference frame, linear acceleration.

Poglavje 1

Uvod

V tej diplomski nalogi smo želeli zajeti človeško gibanje in ga avtomatsko replicirati na humanoidnem robotu. Robot, ki nam je bil na voljo, je NAO podjetja Aldebaran Robotics, ki se uporablja tudi v znanem tekmovanju RoboCup. Obravnavali smo le gibanje roke.

Za namen zajema gibanja smo raziskali več naprav, kot so: Microsoft Kinect, Nintendo Wii MotionPlus ter pametni telefon. Odločili smo se uporabiti pametni telefon, saj vsebuje največ senzorjev ter najbolj obsežen in dokumentiran API za dostop do senzorjev.

Pri zajemu gibanja smo uporabili dva pristopa. Prvi pristop temelji na izračunu linearnega pospeška, iz katerega lahko izračunamo hitrost in pot. Drugi pristop temelji na računalniškem vidu, kjer lahko iz slike direktno izračunamo pozicijo sistema relativno na lego izbranih značilk.

Za replikacijo gibanja smo uporabili metodo diferenčne kinematike, ki temelji na izračunu psevdoinverza Jacobijeve matrike ter posodobitvi trenutne konfiguracije sklepov. Da bi razvili rešitev, robustno na kinematične singularnosti, smo psevdoinverz Jacobijeve matrike izračunali z metodo dušenih najmanjših kvadratov, katera v prostoru sklepov definira regije singularnosti,

v katerih uporablja faktor dušenja, ki je proporcionalem bližini kinematične singularnosti.

V tej diplomski nalogi najprej razložimo delovanje uporabljenih senzorjev. Nato razložimo strojno ter programsko opremo, s katero smo delali. Sledijo metode, katere smo konceptualno razdelili na dva dela. Najprej podamo metode za zajem gibanja, za katere smo razvili tudi lasten API napisan v programskem jeziku Java, ki vsebuje metode za obdelavo signalov ter delo z linearno algebro in kvaternioni. Te metode se izvajajo na pametnem telefonu. Nato podamo metodo za rekonstrukcijo gibanja, ki temelji na diferenčni kinematiki in je odporna na kinematične singularnosti. Na koncu na kratko podamo rezultat in ocenimo uspešnost implementirane metode.

1.1 Sorodno delo

Posnemanje človeškega gibanja z humanoidnim robotom NAO je bilo že večkrat uspešno izvedeno. Večina poskusov temelji na zajemanju gibanja s sistemom Microsoft Kinect, katerega API nudi prepoznavanje človeškega skeleta ter omogoča tridimenzionalno rekonstrukcijo.

V delu [21] sta Fernando Zuher in Roseli Romero uspešno replicirala človeško gibanje s humanoidnim robotom NAO. Za zajem gibanja sta uporabila Microsoft Kinect, s pomočjo katerega sta izračunala kote sklepov uporabnika ter jih prenesla na robota.

V drugem delu [13] sta Jonas Koenemann in Maren Bennewitz iz oddelka računalništvo univerze v Freiburgu uspešno imitirata človekovo gibanje z robotom NAO z drugačnim pristopom. Za zajem gibanja sta uporabila sistem za zajemanje gibanja Xsens MVN, s katerim lahko senzorje pripnemo na človeško telo. Tako sta lahko kote sklepov uporabnika pretvorila v kote sklepov robota NAO.

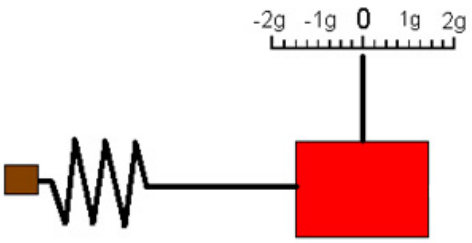
Poglavje 2

Opis senzorjev

Današnji pametni telefoni uporabljajo vrsto senzorjev, kot na primer: žiroskop, pospeškometer, magnetometer, GPS in video kamera. Ti senzorji se uporabljajo predvsem za namene uporabniškega vmesnika, iger ter aplikacije virtualne realnosti, kjer zaznavajo stvari kot so naklon in kretnje. Nekaj teh senzorjev bomo predstavili v tej diplomski nalogi.

2.1 Pospeškometer

Pospeškometer [19] je naprava, ki meri pospešek gibanja strukture. Za razumevanje delovanja si lahko pospeškometer predstavljamo kot maso na vzmeti, kot to prikazuje slika 2.1.



Slika 2.1: Konceptualna shema preprostega pospeškometra z eno prostorsko stopnjo.

Slika 2.1 prikazuje maso, ki lahko prosto drsi po eni osi. Masa je povezana z vzmetjo, ki na maso v staju mirovanja ne deluje z nobeno silo. Če sistem premaknemo v levo, potem vzmet deluje s potrebno silo na maso, da se ta premakne s strukturo. Stanje lahko opišemo z enačbo 2.1

$$ma = k\Delta x, \tag{2.1}$$

kjer m predstavlja konstantno maso, a pospešek, k koeficient vzmeti in Δx razteg vzmeti. To nam omogoča mero pospeška zapisati kot mero raztega vzmeti

$$a = \frac{k}{m}\Delta x.$$

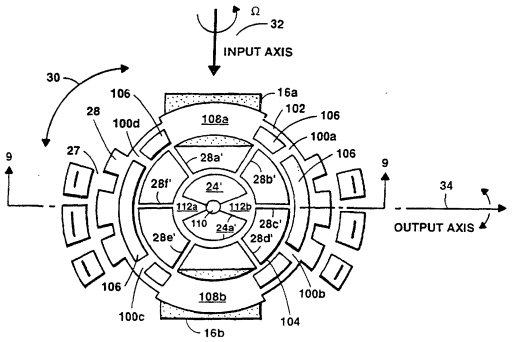
Pri pospešku v obratni smeri velja isti argument, le da se vzmet skrči namesto raztegne.

Moderni pospeškometri seveda ne delujejo po principu mase na vzmeti, ampak sila, ki deluje na strukturo, povzroči, da piezoelektrični material proizvede električni naboj, ki je proporcionalen tej sili. Ker je naboj sorazmeren sili, masa uteži pa konstantna, je naboj sorazmeren pospešku strukture.

2.2 Žiroskop

Žiroskop je naprava za merjenje obodne hitrosti in se uporablja v mnogih navigacijskih sistemih, kot na primer avtopilot ali navigaciji sistem Hubbleovega teleskopa. Osnovni žiroskop temelji na vrteči masi in izkorišča pojav precesije, ki pri pri nagibu osi vrtenja povzroči vrtenje strukture okoli osi pravokotne na os vrtenja in os nagiba. Zaradi ekonomičnosti pametni telefoni uporabljajo tako imenovan Coriolisov vibracijski žiroskop [17].

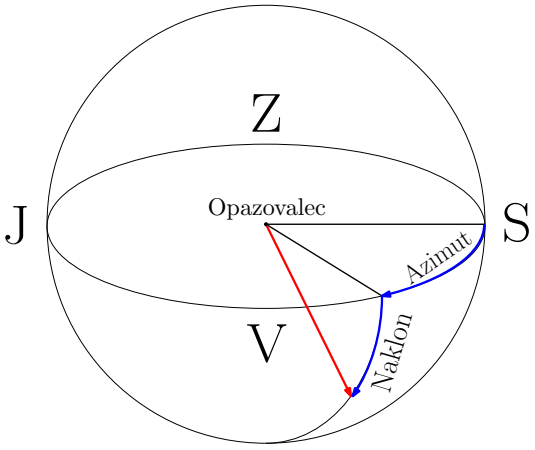
Coriolisov vibracijski žiroskop temelji na vibrirajoči masi, ki pri premiku podpore ohranja ravnino nihanja. Pri tem nastane tako imenovana Coriolisova sila, ki jo je možno izmeriti kapacitivnimi elektrodami.



Slika 2.2: Primer žiroskopa, ki deluje po principu vibrirajočega kolesa.[1]

2.3 Digitalni kompas

Digitalni kompas ali vektorski magnetometer [2] je naprava, s katero lahko izmerimo moč magnetnega polja v določeni smeri, relativno na orientacijo naprave. S ortogonalno postavitvijo treh magnetometrov lahko določimo azimut in naklon, ki sta prikazana na sliki 2.3.



Slika 2.3: Azimut in naklon vektorja magnetnega polja označenega z rdečo barvo.

Če si zemeljsko površino lokalno predstavljamo kot ravnino, potem azimut predstavlja kot med vektorjem magnetnega polja ter vektorjem, ki kaže proti

[1] http://patentimages.storage.googleapis.com/EP0824702B1/00620001.png

severu, obema projeciranima na ravnino zemlje. Naklon pa predstavlja kot med vektorjem magnetnega polja ter ravnino zemlje.

Naprava izkorišča tako imenovan Hallov efekt. Med premikanjem po prevodni plošči izpostavljeni magnetnemu polju na elektrone deluje Lorenzova sila

$$F = q(E + v \times B), \tag{2.2}$$

kjer q predstavlja naboj, E električno polje, v hitrost ter B magnetno polje. Tako lahko pravokotno na smer električnega toka in magnetnega polja zaznamo tako imenovano Hallovo napetost sorazmerno moči magnetnega polja.

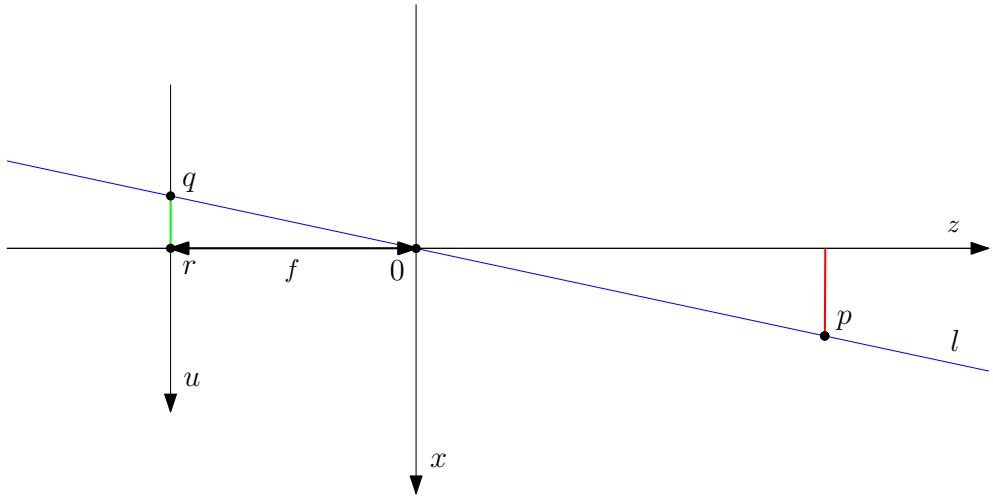
2.4 Video kamera

Matematično video kamero modeliramo z modelom kamere z odprtino [5], ki opisuje razmerje med točko v trodimenzionalnem svetu ter njeno projekcijo na sliki, kjer je zaslon predstavljen s točko in se za zbiranje svetlobe ne uporablja leče. Model je sestavljen iz:

- Trodimenzionalnega ortogonalnega koordinatnega sistema, z osmi x, y in z, v izhodišču katerega se nahaja gorišče. Os z kaže v smeri gledanja kamere.
- Ravnino slike, na katero je projeciran trodimenzionalen svet skozi gorišče. Ravnino slike določata ortogonalni osi u in v, ki sta vzporedni osema x in y ter se nahajata na razdalji f za zaslonom. Razdalji f pravimo tudi goriščna razdalja.
- Točko r, ki se nahaja na preseku ravnine slike z osjo z. To točko poimenujemo središče slike in označimo s c.
- Točko p, ki se nahaja pred zaslonom na koordinatah (x_p, y_p, z_p).
- Projekcijsko premico l točke p, ki gre skozi p in gorišče.

- Projekcijo q točke p na ravnino slike, ki se nahaja na presečišču l z ravnino slike.

Za boljšo predstavo prikazuje slika 2.4 geometrijo modela projecirano na ravnino xz.



Slika 2.4: Geometrija modela luknjaste kamere projecirana na ravnino xz.

Če zelimo izračunati projekcijo točke p na ravnino slike, uporabimo pravilo podobnih trikotnikov, kjer hipotenuza obeh trikotnikov leži na premici l. Tako lahko koordinate točke q zapišemo s formulo 2.3.

$$\begin{pmatrix} u_q \\ v_q \end{pmatrix} = -\frac{f}{z_p} \begin{pmatrix} x_p \\ y_p \end{pmatrix}$$

Model, ki smo ga opisali opisuje, kako delujejo prave kamere z odprtino, ki okolje projecirajo na ravnino slike in nato rotirajo za 180 stopinj. Da dobimo sliko, ki ni rotirana imamo dve možnosti:

- koordinatni sistem ravnine slike rotiramo za 180 stopinj v katerikoli smeri ali

- prestavimo ravnino slike na razdaljo f pred gorišče.

V resničnem svetu zaradi uporabe leč pride do tako imenovanih radialnih ter tangencialnih popačenj [5]. Prav tako ravnina slike ni vedno vzporedna osema x ter y in os z ne seka ravnine slike v središču. Vseh teh problemov se lahko znebimo z uporabo koeficientov radialnega ter tangencialnega popačenja.

Za popravek radialnega popačenja uporabimo formulo

$$\begin{aligned} u_c &= u(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ v_c &= v(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \end{aligned}$$

kjer k_1, k_2 in k_3 predstavljajo koeficiente radialnega popačenja. Za popravek tangencialnega popačenja pa

$$\begin{aligned} u_c &= u + (2p_1 uv + p_2(r^2 + 2u^2)) \\ v_c &= v + (p_1(r^2 + 2v^2) + 2p_2 uv), \end{aligned}$$

kjer p_1 in p_2 predstavljata koeficiente tangencialnega popačenja. Tako bo na popravljeni sliki točka (u, v) imela koordinati (u_c, v_c). Po popravku popačenja lahko koordinate projekcije točke (x, y, z) izračunamo po pravilu 2.3

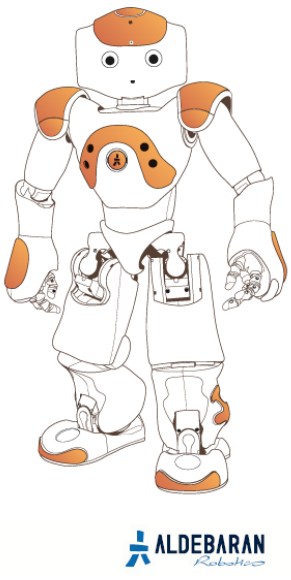
$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = Cp. \tag{2.3}$$

Poglavje 3

Predstavitev opreme

3.1 Humanoidni robot NAO

Diplomsko nalogo smo opravili z uporabo robota NAO Academics Edition [3] podjetja Aldebaran Robotics, ki ga prikazuje slika 3.1.



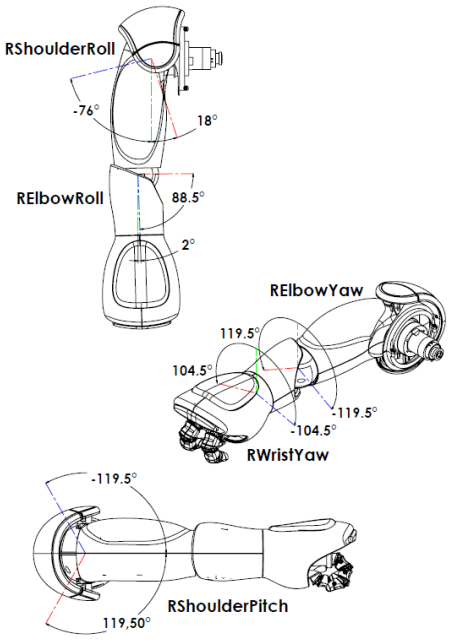
Slika 3.1: Humanoidni robot NAO podjetja Aldeberan Robotics.[1]

[1]https://community.aldebaran-robotics.com/doc/1-14/

NAO Academics Edition je avtonomni humanoidni robot, namenjen predvsem raziskovalnemu delu in ima naslednje tehnične specifikacije:

- 25 prostorskih stopenj, katerih elementi so elektromotorji ter pogoni.
- Senzorsko omrežje, ki ga sestavljajo dve video kameri, štirje mikrofoni, sonar, dva oddajnika in sprejemnika infrardeče svetlobe, osem merilcev tlaka ter devet senzorjev dotika.
- Razne naprave namenjene komuniciranju, kot na primer LED luči ter sintetik zvoka.

NAO vključuje programski tazvojni paket (SDK), ki omogoča razvoj aplikacij v programskih jezikih, kot so C/C++, Python in Java. Za to diplomsko nalogo nas je zanimala predvsem NAO-va roka, prikazana na sliki 3.2.



Slika 3.2: Sklepi roke robota NAO.[2]

[2]http://www.aldebaran-robotics.com/documentation/_images/hardware_rarmjoint_3.3.png

Roka ima pet prostorskih stopenj, od katerih smo za izdelavo diplomske naloge uporabili le štiri: dve rotaciji rame in dve rotaciji komolca, rotacija zapestja pa nas ni zanimala.

3.2 Pametni telefon Samsung Galaxy Note

Telefon Galaxy Note GT-N7000 [1] podjetja Samsung, prikazan na sliki 3.3, je pametni telefon z dvojedrnim 1.4 GHz ARM Cortex A9 procesorjem ter 1 GB pomnilnika. Telefon poganja operacijski sistem Android predstavljen v naslednjem poglavju.

Slika 3.3: Pametni telefon Samsung Galaxy Note GT-N7000.[3]

Telefon Galaxy Note GT-N7000 ima vse senzorje, ki so potrebni za izdelavo diplomske naloge. Vključuje:

- glavno video kamero resolucije 3264x2448 slikovnih točk,

[3]http://www.samsungmobilepress.com/2011/09/01/GALAXY-Note-1

- pospeškometer,
- žiroskop in
- digitalni kompas.

3.3 Operacijski sistem Android

Odprtokodni operacijski sistem Android [16] je podjetje Google leta 2007 razvilo za mobilne naprave z zaslonom na dotik, kot so pametni telefoni in tablični računalniki.

Arhitekturno sistem Android sestavljajo štirje nivoji zloženi v tako imenovani Android sklad, ki vsebuje:

Linux jedro: se nahaja na dnu sklada in skrbi za delovanje strojne opreme, upravljanje s pomnilnikom, varnostne nastavitve, omrežni sklad in ostalo. Ta plast deluje tudi kot abstrakcija med strojno opremo ter programskimi knjižnicami.

Avtohtone knjižnice: to so knjižnice, napisane v programskem jeziku C/C++, ki jih programer lahko kliče preko javanskih vmesnikov. Vsebuje knjižnice kot na primer

- **Upravitelj površine** za skladanje oken na zaslonu,
- **SGL** za dvodimenzionalno grafiko,
- **OpenGL** za trodimenzionalno grafiko,
- **Medijsko ogrodje**, ki podpira predvajanje in snemanje raznih avdio, video in slikovnih formatov, ter
- **SQLite** okrnjena podatkovna baza, prilagojena delovanju na mobilnih napravah.

Na istem nivoju se nahaja tudi izvajalno okolje Android, ki vsebuje javanske knjižnice, katere uporabljamo za pisanje aplikacij. Aplikacije se na sistemu Android poganjajo na navideznem stroju Dalvik (Dalvik VM), ki je vrsta javanskega navideznega stroja (JVM), optimizirana za naprave z malo računske moči in pomilnika. Za razliko od JVM, Dalvik ne poganja datotek s končnico *.class*, temveč detoteke s končnico *.dex*, katere se prevedejo iz datotek tipa *.class* in so približno dvakrat bolj prostorsko učinkovite [16]. Sistem Android hkrati izvaja več instanc Dalvik VM (za vsako aplikacijo eno), kar med drugim omogoča varnost ter izolacijo aplikacij.

Aplikacijsko ogrodje: tu se nahajajo bloki arhitekture, s katerimi aplikacije direktno interaktirajo. Sestavljajo jih programi kot so:

- Upravitelj senzorjev (angl. Sensor Manager),
- Ponudnik vsebin,
- Upravitelj klicev,
- Upravitelj lokacij,
- Upravitelj virov,
- Upravitelj aktivnosti (ang. Activity Manager),

ki skrbijo za osnovne funkcije naprave.

Aplikacijski nivo: to je nivo, s katero ima povprečen uporabnik največ opravka. Tu se nahajajo aplikacije kot so:

- Odjemalec SMS,
- Klicatelj,
- Spletni brskalnik,
- Imenik

in aplikacije ostalih proizvajalcev.

Poglavje 4

Metode

4.1 Računanje orientacije

Pri delu z orientacijo potrebujemo mehanizem za zapis in kompozicijo rotacij. Na voljo imamo več možnosti, kot so na primer: rotacijska matrika, Eulerjevi koti in kvaternioni [20]. Slednje smo uporabili v tej diplomski nalogi, zato jih bomo na kratko predstavili.

4.1.1 Kvaternioni

Kvaternione lahko opišemo kot razširjena kompleksna števila $q = w+ix+jy+kz$, kjer $i^2 = j^2 = k^2 = -1$ in $w, x, y, z \in \mathbb{R}$. Računsko jih lahko predstavimo kot četverke $q = [w, x, y, z] = [w, \vec{v}]$, kjer pravimo, da je w skalarni del, $\vec{v}$ pa vektorski del. Kvaternioni podpirajo naslednje operacije.

Seštevanje: $q_1 + q_2 = [w_1 + w_2, \vec{v_1} + \vec{v_2}]$

Množenje: $q_1 q_2 = [w_1 w_2 - \vec{v_1}\vec{v_2}, \vec{v_1} \times \vec{v_2} + w_1\vec{v_2} + w_2\vec{v_1}]$

Konjunkt: $q^* = [w, -\vec{v}]$

Norma: $\|q\| = \sqrt{qq^*} = \sqrt{w^2 + \vec{v}\vec{v}}$

Inverz: $q^{-1} = \frac{q^*}{\|q\|^2}$

Enotski kvaternioni tvorijo pokritje trodimenzionalnih rotacij. Zajemajo njihovo geometrijo, topologijo in strukturo grupe. Predstavljeni so v obliki

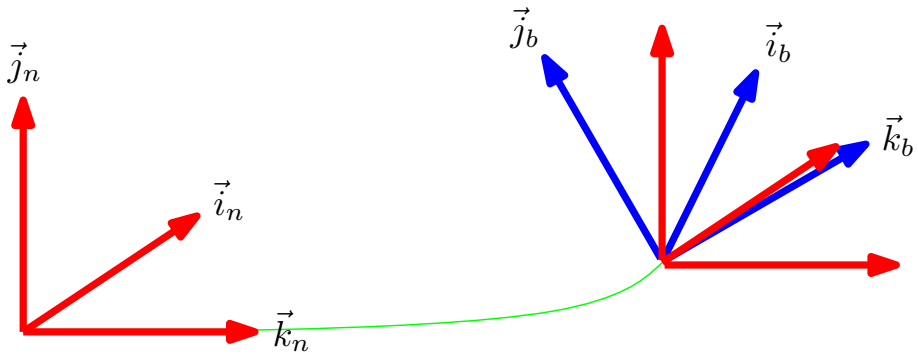
$$q = [w, x, y, z] = \left[\cos\frac{\theta}{2}, \vec{v}\sin\frac{\theta}{2}\right]$$

pri čemer je $\vec{v}$ enotski vektor, ki predstavlja os vrtenja, θ pa kot vrtenja.

Rotacijo trodimenzionalnega vektorja $p = (p_x, p_y, p_z) = ip_x + jp_y + kp_z$ zapišemo kot produkt $p_r = qpq^*$. Kompozicijo rotacij q_1 in q_2 tvorimo s produktom q_2q_1, saj velja $(q_2q_1)p(q_2q_1)^* = q_2(q_1pq_1^*)q_2^*$.

4.1.2 Določanje trenutne orientacije

Pri računanju orientacije je pomemben koncept referenčnega okvirja [20], v katerem dobivamo meritve. V tej diplomski nalogi smo uporabili dva referenčna okvirja prikazana na sliki 4.1. Prvi je okvir telesa (s pripisom b), v katerem dobivamo meritve senzorjev, z izhodiščem v lokaciji sistema in osmi, poravnanimi z osmi sistema. Naslednji okvir je navigacijski okvir (s pripisom n), ki ima prav tako izhodišče v lokaciji sistema, osi pa poravnane s severom, vzhodom ter silo gravitacije.



Slika 4.1: Navigacijski okvir (rdeče) ob zagonu sistema, ter navigacijski okvir in okvir telesa (modro) po premiku sistema.

Pri računanju orientacije moramo izračunati rotacijo r_n, ki slika iz okvira telesa v navigacijski okvir. Da dosežemo željeno odzivnost in dolgoročno

natančnost, se lotimo računanja orientacije na dva načina.

Prvi izračun orientacije temelji na izkoriščanju meritev žiroskopa. Obodna hitrost, ki jo vzorčimo iz žiroskopa, je podana v obliki:

$$\vec{\omega} = \frac{d\vec{\theta}}{dt}. \tag{4.1}$$

Rotacijo r_b, ki predstavlja spremembo orientacije tega časovnega koraka v telesnem okviru konstruiramo v kvaternionski obliki po pravilu 4.2

$$\begin{aligned} \theta &= \|\vec{\omega}\|\Delta t \\ \vec{v} &= \frac{\vec{\omega}}{\|\vec{\omega}\|} \\ r_b &= \left[\cos\frac{\theta}{2}, \vec{v}\sin\frac{\theta}{2}\right]. \end{aligned} \tag{4.2}$$

Če predpostavimo, da je trenutna orientacija r_n pravilna, dobimo novo orientacijo po pravilu $r_n \leftarrow r_n r_b$.

Ker dolgoročni zamik predstavlja velik problem pri uporabi žiroskopov rabimo način, kako žiroskop kalibrirati. To storimo z uporabo digitalnega kompasa in pospeškometra.

Če $\vec{a}_b$ predstavlja stanje pospeškometra in $\vec{m}_b$ stanje kompasa ob zagonu sistema, lahko navigacijski okvir F_n konstruiramo po pravilu:

$$F_n = \begin{pmatrix} \vec{i}_x & \vec{j}_x & \vec{k}_x \\ \vec{i}_y & \vec{j}_y & \vec{k}_y \\ \vec{i}_z & \vec{j}_z & \vec{k}_z \end{pmatrix}.$$

kjer bazne vektorje $\vec{i}, \vec{j}$ ter $\vec{k}$ izračunamo na naslednji način:

$$\begin{aligned} \vec{j} &= \frac{a_b}{\|a_b\|} \\ \vec{k} &= \frac{m_b - proj_{a_b} m_b}{\|m_b - proj_{a_b} m_b\|} \\ \vec{i} &= j \times k. \end{aligned}$$

Za vsako naslednjo meritev po istem pravilu izračunamo telesni okvir F_b, nato orientacijo izračunamo z reševanjem sistema enačb $F_n = RF_b$, kjer za reševanje sistema uporabimo metodo LU razcepa [10]. Tako nam matrika R doloca kvaternion r_n, s katerim lahko popravimo zamik žiroskopa.

4.2 Računanje pozicije

Računanja pozicije smo se lotili z dvema pristopoma. Prvi pristop temelji na računanju linearnega pospeška, medtem ko drugi temelji na računanju pozicije s pomočjo računalniškega vida.

4.2.1 Določanje pozicije s pomočjo pospeškov

Pri računanju pozicije s pomočjo pospeškov [6] si pomagamo s pospeškometrom in trenutno orientacijo. Ta pristop temelji na izračunu linearnega pospeška a_n^l v navigacijskem okviru ter dvojnem numeričnem integriranju, kar nam vrne pot.

Naj bo a_b trenutna meritev pospeškometra. Če želimo izračunati linearni pospešek, moramo a_b preslikati v navigacijski okvir ter ga razstaviti na gravitacijski pospešek ter linearni pospešek, kot je to prikazano v enačbi 4.3

$$\begin{aligned} a_b &= g_b + a_b^l \\ r_n a_b &= r_n g_b + r_n a_b^l \\ a_n &= g_n + a_n^l. \end{aligned} \tag{4.3}$$

Ker je gravitacijski pospešek konstanten, ga lahko preprosto odštejemo:

$$a_n^l = a_n - g_n.$$

Ker velja $a_n^l = \frac{dv_n}{dt} = \frac{d^2 s_n}{dt^2}$ lahko z dvojnim integralom linearnega pospeška do trenutnega časa T izračunamo pot po pravilu 4.4

$$s_n(T) = \int_0^T \int_0^{t_1} a_n^l(t) dt dt_1. \tag{4.4}$$

Pri računanju integrala se lahko poslužujemo različnih numeričnih metod, kot na primer Newton-Catesova pravila (pravokotniško, trapezno, Simpsonovo, itd.).

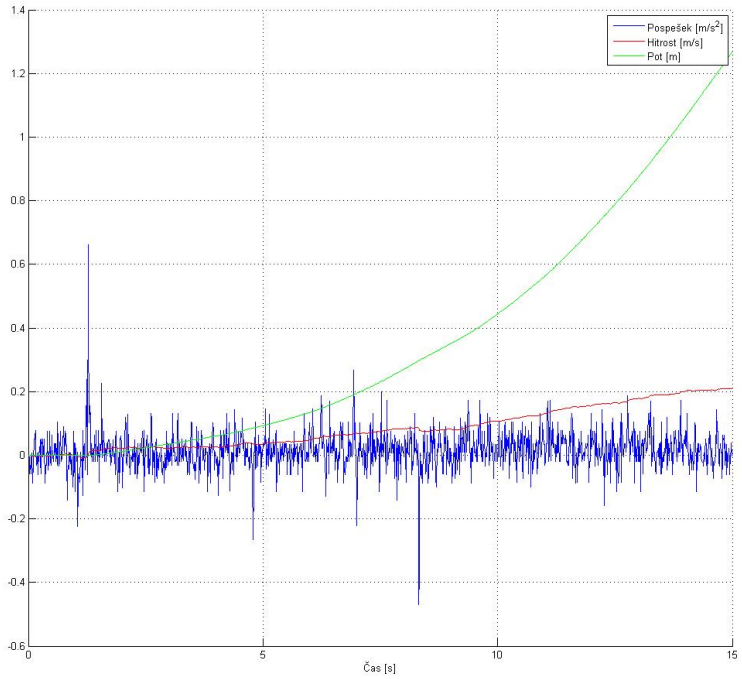
Pri tem pristopu hitro naletimo na težave zaradi napak senzorjev. Slika 4.2 prikazuje y koordinato izračunanega linearnega pospeška, hitrosti ter poti pri mirujočem sistemu.

Vidimo, da se je v petnajstih sekundah akumuliralo skoraj meter in pol napake. Če pogledamo stanje sistema v prvih nekaj minutah, kot prikazuje slika 4.3, opazimo, da se napaka obnaša naključno. V delu [12] je šum pospeškometra modeliran kot mehanski in električni šum, kjer mehanski šum povežejo z Brownovim ter belim šumom, katerih modeliranje je izven obsega te diplomske naloge.

4.2.2 Določanje pozicije s pomočjo računalniskega vida

Pri drugem pristopu smo pozicijo izračunali tako, da smo pred telefom postavili znan vzorec in sledili štirim značilkam tega vzorca.

Za prepoznavanje značilk na sliki, ob inicializaciji sistema shranimo poleg tridimenzionalne pozicije značilke še relativno velik (10×10 pikslov) del slike, ki se nahaja okoli značilke in predstavlja predlogo značilke, s katero iščemo značilko na naslednjih slikah. Med delovanjem sistema predlog ne posodabljamo, saj to vodi do dolgoročnega zamika. Za identifikacijo značilke v novi



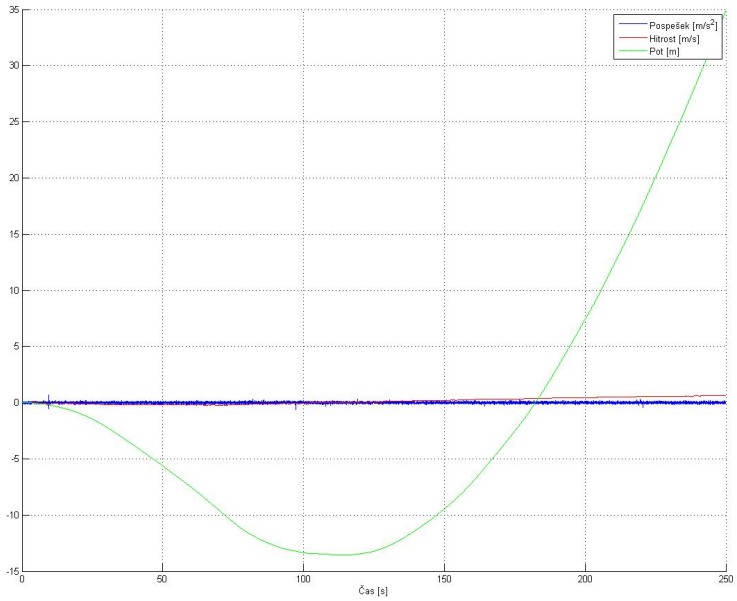
Slika 4.2: Graf prvih petnajstih sekund linearnega pospeška, izračunane hitrosti ter poti po eni koordinati navigacijskega okvira pri mirujočem telefonu.

sliki uporabimo metodo drsečega okna, pri čemer za mero podobnosti uporabimo metodo normalizirane navzkrižne korelacije [4] v regiji, kjer je bila značilka zadnjič opažena:

$$R(x,y) = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} T(i,j)I(x+i,y+j)}{\sqrt{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} T(i,j)^2 \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} I(x+i',y+j')^2}},$$

kjer T predstavlja matriko predloge, I pa matriko slike.

Za določanje pozicije vzorca v telesnem koordinatnem sistemu smo uporabili Levenberg-Marquardtovo [14, 15] optimizacijsko metodo, ki najde takšen položaj vzorca, ki minimizira kvadrat napake projekcije. Napako projekcije



Slika 4.3: Graf prvih nekaj minut linearnega pospeška, izračunane hitrosti in poti po eni koordinati navigacijskega okvira pri mirujočem telefonu.

definiramo kot razdaljo med opaženo ter projicirano značilko.

Če začetne pozicije značilk označimo s $\{p_{0_0}, p_{0_1}, p_{0_2}, p_{0_3}\}$ ter trenutne opažene pozicije v telesnem okviru s $\{p_{b_0}, p_{b_1}, p_{b_2}, p_{b_3}\}$, potem premik od začetne pozicije izračunamo po pravilu:

$$t = \frac{1}{4}\sum_{i=0}^{3}(p_{0_i} - r_n p_{b_i}).$$

Preden predstavimo delovalje Levenberg-Marquardtove metode, moramo predstaviti še Jacobijevo matriko.

Jacobijeva matrika

Jacobijeva matrika je v vektorski analizi matrika vseh parcialnih odvodov prve stopnje.

Naj bo $f : \mathbb{R}^n \to \mathbb{R}^m$ vektorska funkcija, podana s predpisom:

$$f(x_0, x_1, ..., x_{n-1}) = (f_0(x_0, ..., x_{n-1}), f_1(x_0, ..., x_{n-1}), ..., f_{m-1}(x_0, ..., x_{n-1})).$$

Jacobijeva matrika vse parcialne odvode funkcij $f_0, f_1, ..., f_{m-1}$ organizira v matriko $J \in \mathbb{R}^{m \times n}$ z naslednjim predpisom:

$$J = \begin{pmatrix} \frac{df_0}{dx_0} & \frac{df_0}{dx_1} & \cdots & \frac{df_0}{dx_{n-1}} \\ \frac{df_1}{dx_0} & \frac{df_1}{dx_1} & \cdots & \frac{df_1}{dx_{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{df_{m-1}}{dx_0} & \frac{df_{m-1}}{dx_1} & \cdots & \frac{df_{m-1}}{dx_{n-1}} \end{pmatrix}.$$

Pomembna je, saj predstavlja linearni člen razvoja funkcije f v Taylorjvo vrsto okoli točke $(x_0, x_1, ..., x_{n-1})$. To nam omogoča linearno aproksimirati funkcijo f.

Levenberg-Marquardtova optimizacijska metoda

Levenberg-Marquardtova optimizacijska metoda je ena standardnih metod za reševanje nelinearnih problemov najmanjših kvadratov srednje velikosti.

Metoda je kombinacija dveh optimizacijskih metod: gradientne in Gauss-Newtnove [7]. Ko so parametri daleč od optimalnih vrednosti, se metoda obnaša podobno kot gradientna metoda, ko pa so parametri blizu optimalnih vrednosti, pa kot Gauss-Newtnova metoda.

Podatkovne točke $(x_0, y_0), (x_1, y_1), ..., (x_{m-1}, y_{m-1})$ želimo modelirati s funkcijo $y = \hat{y}(x, \theta)$, ki je poleg x odvisna tudi od n parametrov $\theta = (\theta_0, \theta_1, ..., \theta_{n-1})$,

kjer $m \geq n$. Iščemo takšne parametre θ, pri katerih se bo funkcija f podatkovnim točkam pri kriterijski funkciji 4.5 kar najbolje prilegala.

$$E(\theta) = \frac{1}{2}\sum_{i=0}^{m-1}(y_i - \hat{y}(x_i, \theta))^2 = \frac{1}{2}(y - \hat{y}(x, \theta))^T(y - \hat{y}(x, \theta)) \tag{4.5}$$

Kot ostale numerične optimizacijske metode, je Levenberg-Marquardtova metoda iterativna. Uporabnik mora podati začetni približek θ, nato pa se ta v vsaki iteraciji zamenja z novim približkom $\theta + \delta$.

Blizu minimuma metoda aproksimira E, kot kvadratično funkcijo v odvisnosti od θ. To stori tako, da aproksimira $\hat{y}$, kot linearno funkcijo v odvisnosti od θ okoli neke fiksne vrednosti θ_0:

$$\tilde{y}(x, \theta) = \hat{y}(x, \theta_0) + (\theta - \theta_0)^T \nabla \hat{y}(x, \theta_0). \tag{4.6}$$

Dobimo novo kriterijsko funkcijo:

$$\tilde{E}(\theta) = \frac{1}{2}(y - \tilde{y}(x, \theta))^T(y - \tilde{y}(x, \theta))$$

in pa njen gradient:

$$\nabla \tilde{E}(\theta) = (y - \tilde{y}(x, \theta))\nabla \tilde{y}(x, \theta).$$

Če v zadnji enačbi zamenjamo $\tilde{y}$ ter označimo z $d = (\hat{y}(x, \theta_0) - y)\nabla \hat{y}(x, \theta_0)$ in $H = \nabla \hat{y}(x, \theta_0)\nabla \hat{y}(x, \theta_0)^T$ dobimo v minimumu:

$$\nabla \tilde{E}(\theta) = H(\theta - \theta_0) + d = 0$$

oziroma:

$$\theta = \theta_0 - H^{-1}d.$$

Tako dobimo iterativno pravilo 4.7:

$$\theta_{i+1} = \theta_i - H^{-1}d. \tag{4.7}$$

Pravilo ni vedno boljše od gradientne metode, saj predpostavlja, da je $\hat{y}$ linearna glede na θ. Metoda, ki jo je izumil Levenberg in nato izboljšal Marquardt, ti dve pravili meša tako, da se giblje v negativni smeri gradienta, dokler se ne približa minimumu, nato postopoma preklopi na kvadratično pravilo. Podana je s predpisom 4.8:

$$\theta_{i+1} = \theta_i - (H + \lambda \text{diag}(H))^{-1}d. \tag{4.8}$$

Ko je λ majhna, se pravilo približa kvadratični aproksimaciji, ko pa je velika, pa gradientni metodi. Iteracija algoritema izgleda takole:

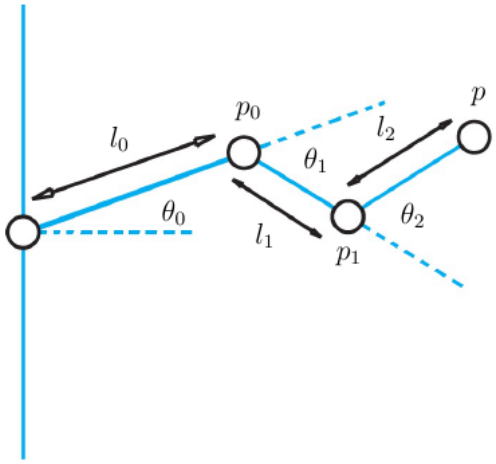
$\theta_{i+1} \leftarrow \theta_i - (H + \lambda \text{diag}(H))^{-1}d$;
$\delta_{i+1} \leftarrow$ oceni napako;
if $\delta_{i+1} \geq \delta_i$ **then**
    $\lambda \leftarrow 10\lambda$;
    ovrzi korak;
else
    $\lambda \leftarrow \frac{\lambda}{10}$;
    $i \leftarrow i + 1$;
end

4.3 Kinematika in inverzna kinematika

Za repliciranje gibov z robotom uporabimo pojma kinematike in inverzne kinematike. Robotovo roko predstavimo kot hierarhično okostje s končnim efektorjem v dlani. Nato lahko s pomočjo kinematike opišemo položaj dlani, z inverzno kinematiko pa premikanje sklepov rame ter komolca glede na spremembo položaja dlani.

4.3.1 Kinematika

Kinematika [11] opisuje premikanje strukture hierarhičnega okostja, glede na premikanje sklepov. Problem si predstavimo takole: pri podanih kotih sklepov $\theta = (\theta_0, \theta_1, ..., \theta_{m-1})$ ter dolžinah povezav $l = (l_0, l_1, ..., l_{m-1})$ želimo izračunati pozicijo končnega efektorja $p = (p_0, p_1, ..., p_{n-1}) = f(\theta)$ v kartezičnih koordinatah. Situacijo prikazuje slika 4.4.



Slika 4.4: Primer hierarhičnega okostja, kjer p predstavlja pozicijo končnega efektorja, p_0 ter p_1 predstavljata poziciji sklepov, l_0, l_1 ter l_2 dolžine povezav in θ_0, θ_1 ter θ_2 kote sklepov.

Če v primeru slike 3.2 označimo kote RShoulderPitch z α, RElbowRoll z β, RShoulderRoll z γ ter RElbowYaw z δ in definiramo vektorja nadlahti z $\vec{a} = (0, 0, a)$ in podlahti z $\vec{b} = (0, 0, b)$ potem lahko pozicijo dlani izračunamo z enačbo 4.9:

$$\vec{p} = R_a(\vec{a} + R_b\vec{b}), \tag{4.9}$$

kjer sta $R_a = R_\gamma R_\alpha$ in $R_b = R_\delta R_\beta$. $R_\alpha, R_\beta, R_\gamma$ in R_δ pa so Eulerjevi koti določeni z 4.10

$$R_\alpha = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \qquad R_\beta = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \tag{4.10}$$
$$R_\gamma = \begin{pmatrix} \cos\gamma & 0 & \sin\gamma \\ 0 & 1 & 0 \\ -\sin\gamma & 0 & \cos\gamma \end{pmatrix} \qquad R_\delta = \begin{pmatrix} \cos\delta & -\sin\delta & 0 \\ \sin\delta & \cos\delta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

4.3.2 Singularni razcep

Singularni razcep (SVD) [10] je faktorizacija matrike $A \in \mathbb{R}^{m\times n}$ podan s predpisom:

$$A = U\Sigma V^T$$

kjer sta matriki $U \in \mathbb{R}^{m\times n}$ in $V \in \mathbb{R}^{n\times n}$ ortogonalni, $\Sigma \in \mathbb{R}^{n\times n}$ pa diagonalna z vrednostmi $\sigma_0, \sigma_1, ..., \sigma_{n-1}$. Stolpci $u_0, u_1, ..., u_{n-1}$ matrike U se imenujejo levi singularni vektorji, stolpci $v_0, v_1, ..., v_{n-1}$ matrike V pa desni singularni vektorji. Elementi $\sigma_0, \sigma_1, ..., \sigma_{n-1}$ matrike Σ se imenujejo singularne vrednosti. Če je $m < n$ je SVD matrike A definiran z singularnim razcepom matrike A^T.

V tej diplomski nalogi nas zanimata predvsem izreka 4.1 in 4.2.

Izrek 4.1. *Naj bodo $\sigma_0 \geq \sigma_1 \geq ... \geq \sigma_{k-1} > \sigma_k = \sigma_{k+1} = ... = 0$ singularne vrednosti matrike A. Potem je rang matrike A enak k.*

Izrek 4.2. *Če ima matrika A poln rang, potem ima naloga $\min_x \|b - Ax\|_2$ rešitev:*

$$x = V\Sigma^{-1}U^T b.$$

Tako lahko z uporabo SVD lahko izračunamo tudi tako imenovano Moore-Penroseovo psevdoinverzo matriko A^+ matrike A po pravilu:

$$A^+ = (A^T A)^{-1} A^T = V \Sigma^{-1} U^T.$$

Moore-Penroseova psevdoinverzna matrika se po navadi uporablja za izračun najboljše rešitve (po metodi najmanjših kvadratov) sistema enačb z več rešitvami.

4.3.3 Inverzna kinematika

Kot ime namiguje, se inverzna kinematika [11] ukvarja z obratnim problemom kot kinematika in sicer nas zanimajo vrednosti kotov sklepov $\theta = (\theta_0, \theta_1, ..., \theta_{m-1})$ pri dani pozicijo končnega efektorja $p = (p_0, p_1, ..., p_{n-1})$. Torej iščemo funkcijo 4.11

$$f^{-1}(p) = \theta. \tag{4.11}$$

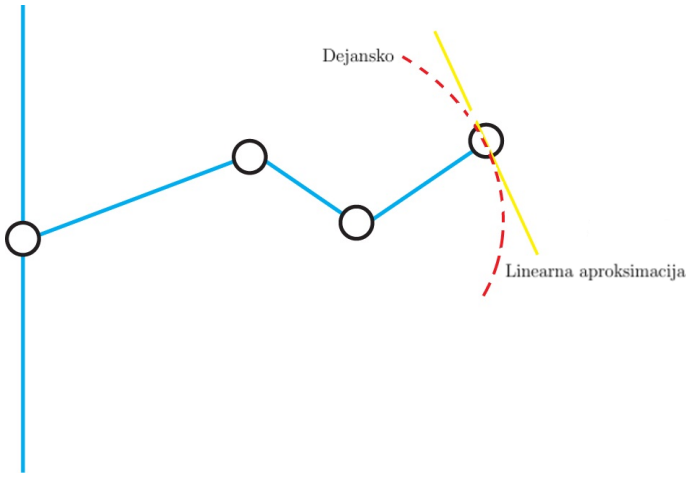
Pri inverzni kinematiki se pojavijo dodatne težave, kot na primer več možnih rešitev ali pa nobene rešitve sistema enačb 4.11. Prav tako je izračun sistema 4.11 računsko veliko bolj zahteven. S standardnim pristopom funkcijo f linearno aproksimiramo okoli trenutne konfiguracije θ_0, kot je izpeljano v 4.12:

$$\begin{aligned} f(\theta_1) &= f(\theta_0) + J(\theta_1 - \theta_0) \\ p_1 - p_0 &= J(\theta_1 - \theta_0) \\ \Delta p &= J \Delta \theta, \end{aligned} \tag{4.12}$$

kjer J predstavlja Jacobijevo matriko funkcije f. Tako lahko izračunamo razliko v kotih sklepov po pravilu 4.13. Situacijo prikazuje slika 4.5:

$$\Delta \theta = J^{-1} \Delta p. \tag{4.13}$$

Zaradi boljše natančnosti smo pri računanju Jacobijeve matrike poziciji dodali še kot φ, ki predstavlja rotacijo dlani okoli lastne osi z. Tako je Jacobijeva matrika po stolpcih podana z naslednjim predpisom 4.14:



Slika 4.5: Linearna aproksimacija funkcije f okoli trenutne konfiguracije θ_0.

$$\frac{df}{d\alpha} = \begin{pmatrix} \sin\gamma(b\cos\alpha\sin\beta\sin\delta - \sin\alpha(a + b\cos\beta)) \\ -\cos\alpha(a + b\cos\beta) - b\sin\alpha\sin\beta\sin\delta \\ \cos\gamma(b\cos\alpha\sin\beta\sin\delta - \sin\alpha(a + b\cos\beta)) \\ 0 \end{pmatrix} \tag{4.14}$$

$$\frac{df}{d\beta} = \begin{pmatrix} b(\cos\beta(\cos\gamma\cos\delta + \cos\alpha\sin\gamma\sin\delta) - \cos\alpha\sin\beta\sin\gamma \\ b(\sin\alpha\sin\beta + \cos\alpha\cos\beta\sin\delta) \\ -b(\cos\alpha\sin\beta\cos\gamma + \cos\beta(\sin\gamma\cos\delta - \sin\alpha\cos\gamma\sin\delta)) \\ 0 \end{pmatrix}$$

$$\frac{df}{d\gamma} = \begin{pmatrix} \cos\alpha\cos\gamma(a + b\cos\beta) + b\sin\beta(\sin\alpha\cos\gamma\sin\delta - \sin\gamma\cos\delta) \\ 0 \\ -\cos\alpha\sin\gamma(a + b\cos\beta) - b\sin\beta(\cos\gamma\cos\delta + \sin\alpha\sin\gamma\sin\delta) \\ 0 \end{pmatrix}$$

$$\frac{df}{d\delta} = \begin{pmatrix} b\sin\beta(\sin\alpha\sin\gamma\cos\delta - \cos\gamma\sin\delta) \\ b\cos\alpha\sin\beta\cos\delta \\ b\sin\beta(\sin\alpha\cos\gamma\cos\delta + \sin\gamma\sin\delta) \\ -1 \end{pmatrix}.$$

Kinematične singularnosti

Kinematična singularnost se zgodi, ko za neko konfiguracijo θ postane determinanta Jacobijeve matrike ničelna in ne moremo izračunati inverza J^{-1}.

Za rešitev, ki je robustna na kinematične singularnosti, smo se obrnili na rešitev [8], ki izračuna psevdoinverz Jacobijeve matrike z metodo dušenih najmanjših kvadratov:

$$J^* = J^T(JJ^T + \Lambda)^{-1} = \sum_{k=0}^{3} \frac{\sigma_k}{\sigma_k^2 + \lambda_{k,k}^2} v_k u_k^T,$$

kjer je Λ diagonalna matrika z elementi $\lambda_{k,k}^2$ in predstavlja faktor dušenja, σ_i k-ta singularna vrednost, ter u_k in v_k k-ti levi in desni singularni vektor. Ta pristop nam zagotavlja zveznost in dobro pogojenost po celotnem prostoru manipulatorja na račun večje napake rekonstrukcije zaradi dušenja.

Napako, ki se pojavi zaradi dušenja lahko zmanjšamo tako, da zmanjšamo faktor dušenja, ko se končni efektor nahaja daleč od kinematičnih singularnosti. V ta namen definiramo regije singularnosti v bližini kinematičnih singularnosti tako, da je Jacobijeva matrika polnega ranga izven teh regij in se lahko uporabi ničelni faktor dušenja. Faktor dušenja v teh regijah je proporcionalen bližini kinematični singularnosti. Tako faktor dušenja določimo na naslednji način:

$$\lambda_{k,k}^2 = \begin{cases} 0 & \sigma_k \geq \epsilon_k \\ \left(1 - \left(\frac{\sigma_k}{\epsilon_k}\right)^2\right) \lambda_{max} & \sigma_k < \epsilon_k \end{cases} \tag{4.15}$$

kjer je σ_k k-ta singularna vrednost Jacobijeve matrike, λ_{max} ter ϵ_k pa sta empirično določena parametra.

Poglavje 5

Rezultati

Da bi ocenili uspešnost delovanja sistema, smo izvedli tri eksperimente. Prvi eksperiment je bil namenjen oceni natančnosti izračuna orientacije, drugi oceni natančnosti izračuna pozicije, tretji pa oceni natančnosti repliciranja gibanja. V tem poglavju bomo prikazali rezultate vsakega od teh treh eksperimentov.

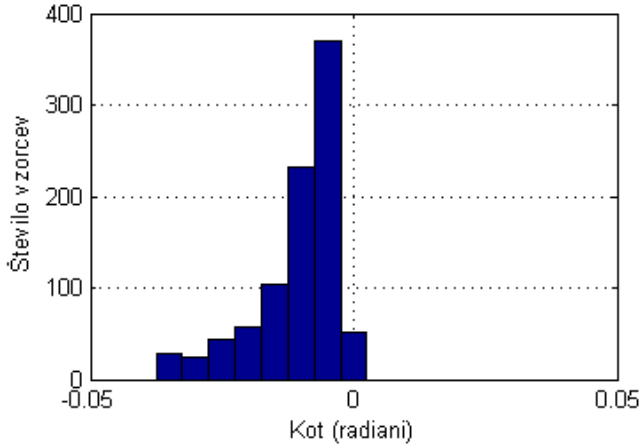
5.1 Rezultati izračuna orientacije

Za oceno natančnosti sistema za izračun orientacije smo izvedli naslednji eksperiment: pametni telefon smo položili na mizo in pustili, da se sistem za računanje orientacije, opisan v poglavju 4.1 inicializira. Nato smo telefon dvignili in nekajkrat naključno zavrteli ter postavili nazaj na mizo v začetni položaj. Na koncu smo iz kvaterniona, ki sistemu določa preslikavo v navigacijski okvir, razbrali kot vrtenja.

Porazdelitev razbranega kota smo modelirali kot Gaussovo porazdelitev $\hat{\varphi} \sim N(\mu, \sigma)$, kjer sta parametra porazdelitve podana v enačbi 5.1

$$\begin{aligned} \mu &= -0.0105 \\ \sigma &= 0.0082. \end{aligned} \tag{5.1}$$

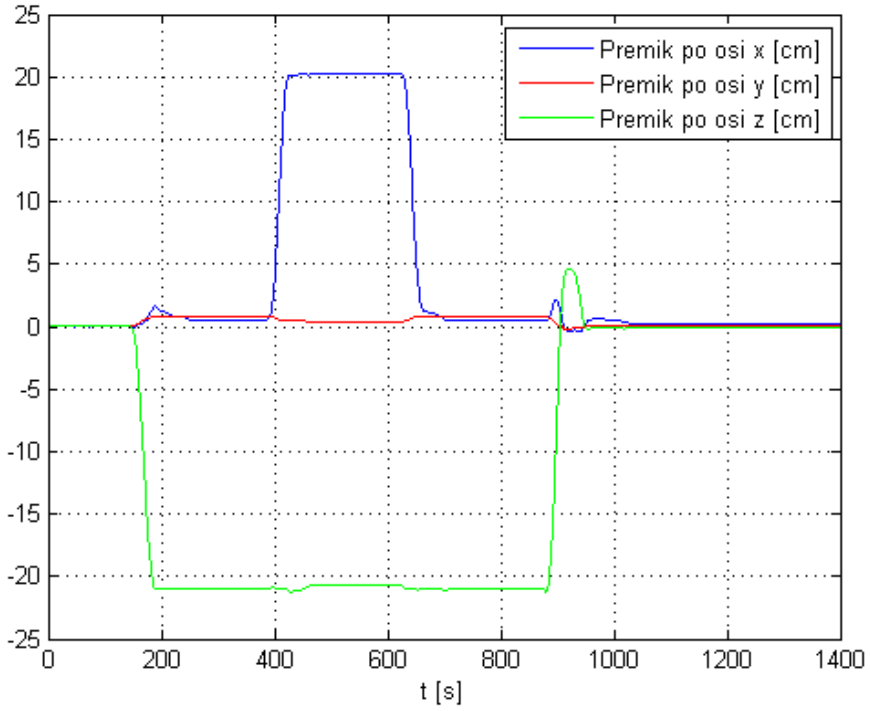
Največja napaka kota med pravilno in izračunano orientacijo znaša $\hat{e}_{max} = 0.0354$ radianov. Porazdelitev razbranega kota je prikazana na sliki 5.1.



Slika 5.1: Porazdelitev kota med pravilno in izračunano orientacijo.

5.2 Rezultati izračuna pozicije

Da bi ocenili natančnost sistema za izračun pozicije, smo pametni telefon postavili pred vnaprej poznan vzorec ter pustili sistemu za izračun pozicije, da se inicializira. Nato smo telefon premaknili dvajset centimetrov v negativni smeri osi z in nato dvajset centimetrov v pozitivni smeri osi x, ga tam nekaj časa pustili in premaknili nazaj v začetni položaj. Graf premika po vsaki izmed osi prikazuje slika 5.2.

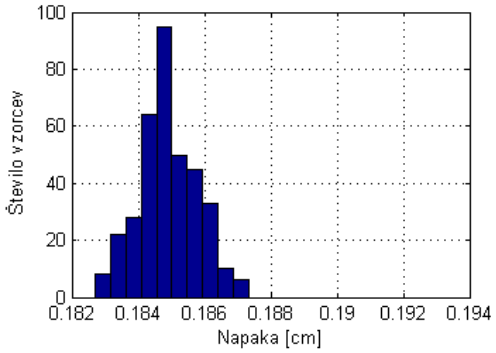


Slika 5.2: Graf meritev premika sistema deset centimetrov v smeri z osi.

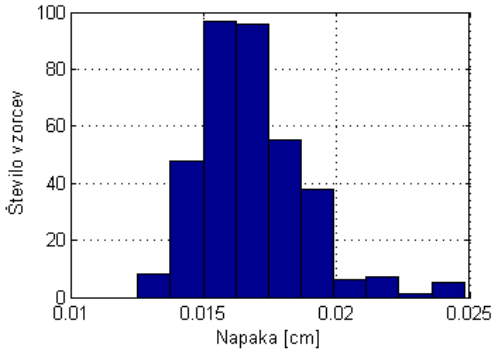
Delovanja sistema nismo ocenjevali v času premikanja, ampak le, ko je naprava bila v stanju mirovanja. Napako sistema po premiku nazaj v začetno pozicijo in v premaknjeni poziciji smo obravnavali posebej. Obe smo modelirali z Gaussovo porazdelitvijo. V začetni poziciji smo oceno porazdelitve napake označili z $\hat{e}_0 \sim N(\vec{\mu}_0, \Sigma_0)$, kjer sta parametra $\vec{\mu}_0$ in Σ_0 podana v enačbi 5.2

$$\begin{aligned} \vec{\mu}_0 &= \begin{pmatrix} 0.1849 \\ 0.0168 \\ -0.1110 \end{pmatrix} \\ \Sigma_0 &= \begin{pmatrix} 7.7678 \times 10^{-7} & 1.2196 \times 10^{-6} & 4.5681 \times 10^{-8} \\ 1.2196 \times 10^{-6} & 3.8016 \times 10^{-6} & 2.2204 \times 10^{-7} \\ 4.5681 \times 10^{-8} & 2.2204 \times 10^{-7} & 1.4663 \times 10^{-8} \end{pmatrix}. \end{aligned} \tag{5.2}$$

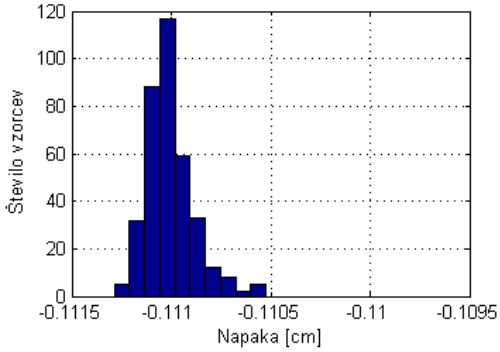
Porazdelitev napake sistema v začetni poziciji prikazuje slika 5.3.



(a) Porazdelitev napake izračunane pozicije v začetnem položaju po osi x.



(b) Porazdelitev napake izračunane pozicije v začetnem položaju po osi y.



(c) Porazdelitev napake izračunane pozicije v začetnem položaju po osi z.

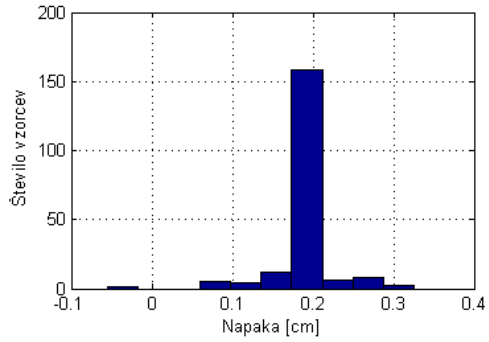
Slika 5.3: Porazdelitve napake izračunane pozicije v začetnem položaju po oseh x 5.3a, y 5.3b in z 5.3c.

V premaknjeni poziciji smo oceno porazdelitve meritev označili z $\hat{e}_1 \sim N(\vec{\mu}_1, \Sigma_1)$. Parametra $\vec{\mu}_1$ in Σ_1 sta podana v enačbi 5.3.

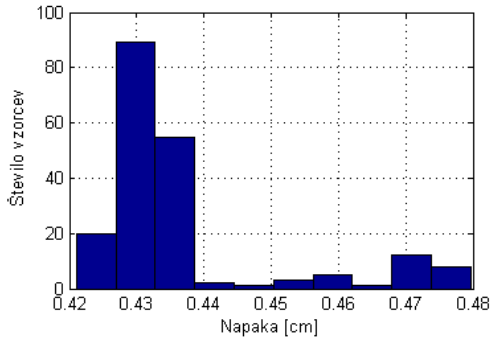
$$\vec{\mu}_1 = \begin{pmatrix} 20.1813 \\ 0.4366 \\ -20.8094 \end{pmatrix} \tag{5.3}$$

$$\Sigma_1 = \begin{pmatrix} 0.0012 & -1.1482 & 6.43 \times 10^{-4} \\ -1.1482 \times 10^{-4} & 2.0201 \times 10^{-4} & -0.002 \\ 6.43 \times 10^{-4} & -0.002 & 0.0225 \end{pmatrix}.$$

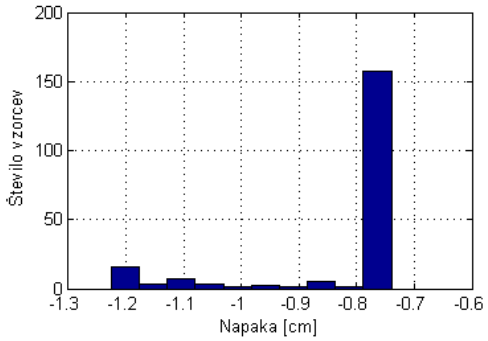
Porazdelitev napake izračunane pozicije po premiku telefona po vsaki izmed osi prikazuje slika 5.4.



(a) Porazdelitev napake izračunane pozicije po osi x po premiku telefona.



(b) Porazdelitev napake izračunane pozicije po osi y po premiku telefona.



(c) Porazdelitev napake izračunane pozicije po osi z po premiku telefona.

Slika 5.4: Porazdelitve napake izračunane pozicije po oseh x 5.4a, y 5.4b in z 5.4c po premiku telefona.

Izračun pokaže, da povprečna absolutna napaka izračunanega premika Δs znaša $\Delta s = 0.9374$ centimetra, največja absolutna napaka pa znaša 1.3194 centimetra. Relativna napaka znaša 3.31%.

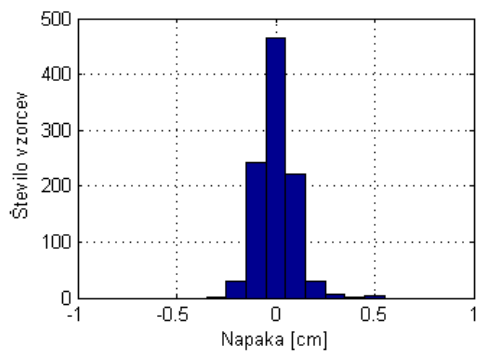
5.3 Rezultati replikacije gibanja

Da bi ocenili natančnost repliciranja gibanja, smo pri prostem gibanju roke ocenili razliko med koordinatami poslanimi sistemu za repliciranje gibanja ter koordinatami robotove dlani po posodobljeni konfiguraciji sklepov.

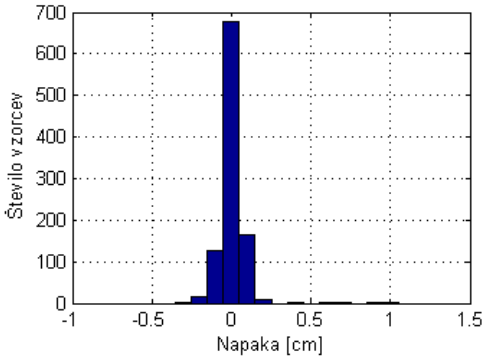
Oceno napake $\hat{e}$ smo modelirali z Gaussovo porazdelitvijo $\hat{e} \sim N(\vec{\mu}, \Sigma)$, kjer so vrednosti parametrov porazdelitve prikazane v enačbi 5.4

$$\begin{aligned} \vec{\mu} &= \begin{pmatrix} 0.0015 \\ 0.0045 \\ 0.0079 \end{pmatrix} \\ \Sigma &= \begin{pmatrix} 0.0077 & 0.0032 & 0.0019 \\ 0.0032 & 0.0067 & 0.002 \\ 0.0019 & 0.002 & 0.0056 \end{pmatrix}. \end{aligned} \tag{5.4}$$

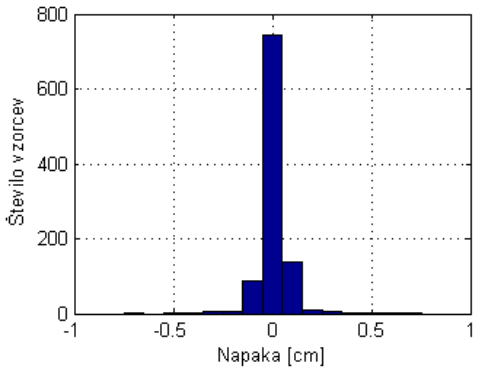
Norma največje napake $\hat{e}_{max}$ znaša $\|\hat{e}_{max}\|_2 = 1.3070$ centimetrov. Porazdelitev napake po vsaki izmed osi prikazuje slika 5.5.



(a) Porazdelitev napake replikacije gibanja po osi x.



(b) Porazdelitev napake replikacije gibanja po osi y.



(c) Porazdelitev napake replikacije gibanja po osi z.

Slika 5.5: Porazdelitve napake sistema za repliciranje gibanja po oseh x 5.5a, y 5.5b in z 5.5c.

5.4 Primerjava rezultatov

Primerjavo rezultatov smo izvedli posebej za izračun pozicije, orientacije ter replikacijo gibanja. Za vsakega od teh bomo podali rezulate v tem poglavju.

5.4.1 Primerjava izračuna orientacije

Metodo izračuna orientacije smo primerjali z metodo uporabljeno v delu [18]. V tem delu so izmerili napako orientacije po enem obratu sistema za 360 stopinj. Po uporabi kompenzacijske funkcije postane natančnost njihovega

sistema primerljiva z našim.

5.4.2 Primerjava izračuna pozicije

Metodo izračuna pozicije smo primerjali z metodo uporabljeno v delu [9], kjer so za izračun pozicije sistema zgradili probabilističen skopi zemljevid okolice. V tem delu so pri premikanju sistema po dimenzij $100cm \times 50cm \times 62cm$ zabeležili povprečno absolutno napako 5.31 centimetrov oziroma relativno napako 5.2% kar je slabše od relativne napake 3.31%, ki smo jo zabeležili v razdelku 5.2.

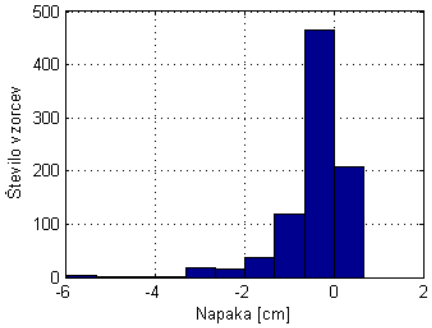
5.4.3 Primerjava replikacije gibanja

Metodo replikacije gibanja smo primerjali z metodo izračuna Moore-Penroseovega prevdoinverza Jacobijeve matrike. V ta namen smo ponovili eksperiment predstavljen v razdelku 5.3, pri čemer smo izračunali psevdoinverz Jacobijeve matrike z Moore-Penroseovo metodo. Tu velja poudariti, da metodi izven regij singularnosti delujeta identično. Porazdelitev napake smo prav tako modelirali z Gaussovo porazdelitvijo, kjer so parametri prikazani v enačbi 5.5.

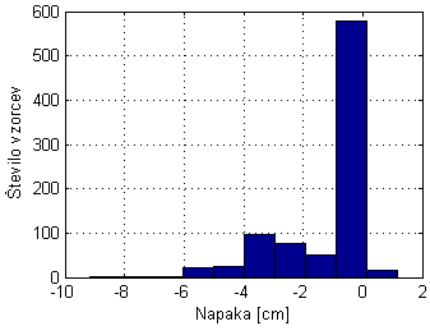
$$\begin{aligned} \vec{\mu} &= \begin{pmatrix} -0.3742 \\ -1.0072 \\ 0.3584 \end{pmatrix} \\ \Sigma &= \begin{pmatrix} 0.6473 & 0.6663 & -0.4247 \\ 0.6663 & 2.6288 & -0.8199 \\ -0.4247 & -0.8199 & 0.4990 \end{pmatrix}. \end{aligned} \tag{5.5}$$

Norma največje napake $\hat{e}_{max}$ v tem primeru znaša 9.6650 centimetrov. Izračun pokaže, da smo povprečno napako zmanjšali iz 1.1326 centimetra na 0.0092 centimetra. Največjo napako pa smo izboljšali za 8.358 centimetrov. Izboljšava je zlasti opazna, ko se konfiguracija sklepov nahaja v bližini kine-

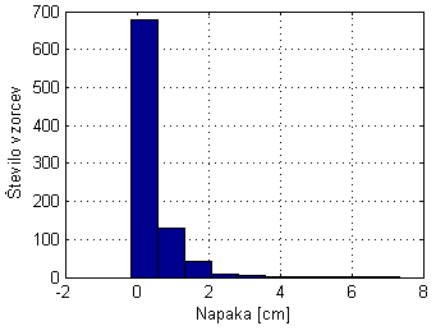
matičnih singularnosti. Slika 5.6 prikazuje porazdelitev napake pri ponovljenem eksperimentu z izračunom prevdoinverza Jacobijeve matrike z uporabo Moore-Penroseove metode.



(a) Porazdelitev napake replikacije gibanja, pri izračunu psevdoinverza Jacobijeve matrike z Moore-Penroseovo metodo, po osi x.



(b) Porazdelitev napake replikacije gibanja, pri izračunu psevdoinverza Jacobijeve matrike z Moore-Penroseovo metodo, po osi y.



(c) Porazdelitev napake replikacije gibanja, pri izračunu psevdoinverza Jacobijeve matrike z Moore-Penroseovo metodo, po osi z.

Slika 5.6: Porazdelitve napake sistema za repliciranje gibanja, pri izračunu psevdoinverza Jacobijeve matrike z Moore-Penroseovo metodo, po oseh x 5.6a, y 5.6b in z 5.6c.

Poglavje 6

Zakjluček

V diplomski nalogi smo raziskali in implementirali pristop zajema ter repliciranja gibov človekove roke s pomočjo pametnega telefona Samsung Galaxy Note ter humanoidnega robota NAO podjetja Aldebaran Robotics.

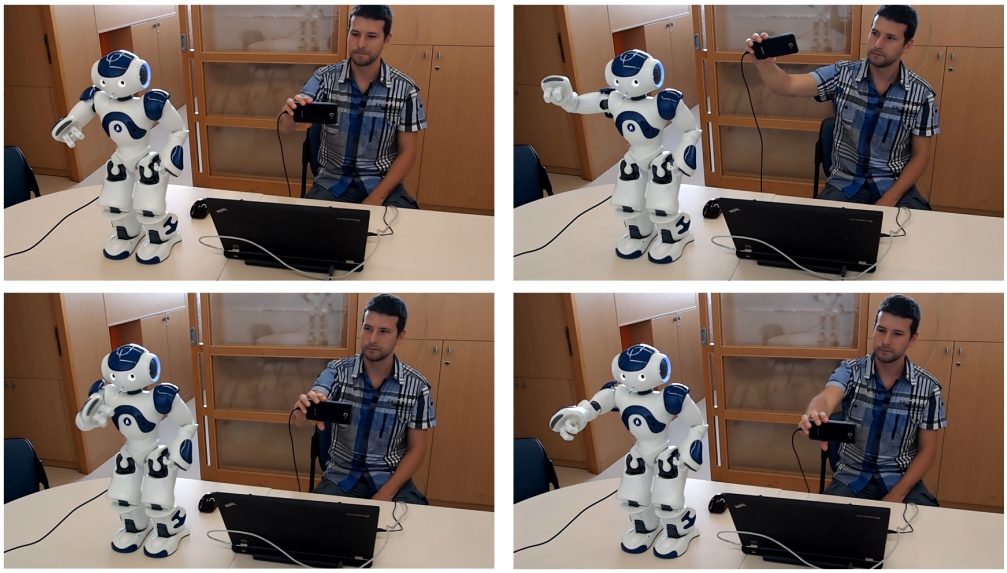
Pri zajemu gibanja roke smo razvili program, ki računa orientacijo in pozicijo dlani. Za določanje orientacije dlani smo uporabili dva pristopa. Prvi pristop temelji na izračunu orientacije iz meritev žiroskopa, kjer iz meritev izračunamo rotacijo časovnega koraka in posodobimo trenutno orientacijo. Drugi pristop temelji na izračunu orientacije iz meritev magnetometra in pospeškometra. V tem pristopu izračunamo referenčni okvir, ter z reševanjem sistema enačb trenutno orientacijo. Za večjo natančnost določimo orientacijo s kombinacijo obeh pristopov.

Pri računanju pozicije dlani smo implementirali dva pristopa. Prvi pristop temelji na izračunu linearnega pospeška iz meritev pospeškometra. To stori tako, da meritev preslika v navigacijski okvir in odšteje gravitacijo, nato z dvokratnim numeričnim integriranjem izračuna pozicijo relativno začetni poziciji. Pristop se je žal izkazal za neuspešnega. Drugi pristop temelji na računanju pozicije s pomočjo računalniškega vida. To smo storili tako, da smo sledili štirim značilkam vzorca, ki smo ga postavili pred video kamero

naprave. Ker smo poznali dimenzije vzorca, smo lahko iz vsake naslednje slike izračunali odmik od začetne pozicije.

Pri repliciranju gibanja smo uporabili metodo diferenčne kinematike, ki razliki pozicije končnega efektorja priredi razliko kotov sklepov okostja, kar stori z računanjem psevdoinverza Jacobijeve matrike. Da bi razvili rešitev, ki je robustna na kinematične singularnosti, smo implementirali pristop, ki izračuna psevdoinverz po metodi dušenih najmanjših kvadratov, kjer smo faktorje dušenja določili empirično.

Slika 6.1 prikazuje utrinke posnetka delovanja sistema.



Slika 6.1: Utrinki posnetka delovanja sistema razvitega v tej diplomski nalogi.

6.1 Možne uporabe

Vodenje robota na daljavo ima veliko možnih uporab. Z nekaj izboljšavami in prilagoditvami, bi lahko sistem uporabili v človeku nevarnih okoljih, kjer bi na primer namesto gasilca v gorečo stavbo poslali robota, katerega bi iz

varne lokacije upravljal strokovnjak.

Sistem bi lahko uporabili tudi kot dodatek robotom, ki sintetizirajo govor. Tako bi ti z uporabo vnaprej posnetih kretenj med sintezo zvoka namigovali lastnosti, kot so velikost, dvom in zanikanje in tako postali človeškim poslušalcem veliko bolj zanimivi in sledljivi.

6.2 Možne razširitve

Pri morebitnih nadaljnih razširitvah sistema bi lahko izboljšati mehanizem zajemanja gibanja. Želeli bi, da sistem ne potrebuje vnaprej poznanega vzorca za računanje odmika od začetne pozicije. Eden od možnih pristopov bi bil pristop iz [9], ki s pomočjo nizkoresolucijske kamere zgradi probabilističen skopi zemljevid okolice in tako doseže lokalizacijo v realnem času. V tem primeru sistem ne bi moral biti lociran pred vnaprej poznanim vzorcem.

Literatura

[1] Galaxy Note GT-N7000 Tech Specs. Dostopno na http://www.samsung.com/uk/support/model/GT-N7000ZBAXEU-techspecs.

[2] Hall effect Magnetometer. Dostopno na http://wikid.eu/index.php/Hall_effect_Magnetometer.

[3] NAO Software 1.14.3 documentation. Dostopno na http://www.aldebaran-robotics.com/documentation/.

[4] Template Matching. Dostopno na http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html.

[5] Camera Calibration and 3D Reconstruction, 2009. Dostopno na http://opencv.willowgarage.com/documentation/camera_calibration_and_3d_reconstruction.html.

[6] Michael Asher in Emmanuel Malikides. IMU Filter. Tehnicno porocilo, ANU College of Engineering & Computer Science, 2011.

[7] Stephen Boyd in Lieven Vandenberghe. *Convex Optimization*. Cambridge University press, Cambridge, 2004.

[8] Stefano Chiaverini. Singularity-Robust Task-Priority Redundancy Resolution for Real-Time Kinematic Control of Robot Manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, 1997.

[9] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, in Oliver Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analisys and Machine Intelligence*, 29(6):1–16, 2007.

[10] James W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Berkeley, 1997.

[11] R. Kelly, V. Santibáñez, in A. Loría. *Control of Robot Manipulators in Joint Space*. Springer, 2005.

[12] Haluk Külah, Junseok Chae, Navid Yazdi, in Khalil Najafi. Noise Analisys and Characterization of a Sigma-Delta Capacitive Microaccelerometer. *IEEE Journal of Solid-State Circuits*, 41(2):352–361, 2006.

[13] Jonas Koenemann in Maren Bennewitz. Whole-Body Imitation of Human Motions with a Nao Humanoid. Objavljeno v *HRI'12 International Conference on Human-Robot Interaction*, stran 500, 2012.

[14] Kenneth Levenberg. A method for the solution of certain problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[15] Donald W. Marquard. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[16] Ziguard Mednieks, Laird Dornin, G.Blake Meike, in Masumi Nakamura. *Programming Android*. O'Reilly Media, Sebastopol, 2012.

[17] Steven Nasiri. A Critical Review of MEMS Gyroscopes Technology and Commercialization Status. Dostopno na `http://www.invensense.com/mems/gyro/documents/whitepapers/MEMSGyroComp.pdf`.

[18] Lauro Ojeda, Hakyoung Chung, in Johann Borenstein. Precision-calibration of Fiber-optics Gyroscopes for Mobile Robot Navigation. Objavljeno v *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, strani 2064–2069, San Francisco, 2000.

[19] Vijayakumar S., Vijila G., Alagappan M., in Anju Gupta. Design and Analysis of 3D Capacitive Accelerometer for Automotive Applications. Objavljeno v *2011 COMSOL Conference*, Bangalore, 2011.

[20] David H. Titterton in John L. Weston. *Strapdown Internal Navigation Technology*. The Institution of Electrical Engineers, The American Institute of Aeronautics and Astronautics, Stevenage, 2004.

[21] Fernando Zuher in Roseli Romero. Recognition of human motions for imitation and control of a humanoid robot. Objavljeno v *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, strani 190–195, 2012.