Jussi Nikula

Kompassin toteuttaminen mobiililaitteeseen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Maanmittaustekniikka

Insinöörityö

31.5.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Jussi Nikula Kompassin toteuttaminen mobiililaitteeseen 24 sivua + 2 liitettä 31.5.2018
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	maanmittaustekniikka
Ohjaaja	lehtori Jussi Laari

Modernit mobiililaitteet sisältävät anturit magnetismille ja kiihtyvyydelle. Näiden tiedoista voidaan muodostaa kompassisuunta. Asiaa käsittelevät verkkolähteet ovat vaihtelevan tasoisia, joko sisältäen raskasta matemaattista teoriaa vähäisellä yleisesittelyllä, tai ollen vain hyvin kevyitä esityksiä. Tämä insinöörityö pyrki antamaan kompassisovelluksen toteuttamisesta mobiililaitteeseen ymmärrettävän esittelyn, jonka avulla voidaan muodostaa yleiskäsitys aiheesta ja ymmärtää paremmin laskennan periaatteita ja käytäntöä.

Piirivalmistajan julkaisemia kompassilaskentaa ja kalibrointia kuvaavia dokumentteja esiteltiin ja niiden sisältämää laskentaa kuvailtiin havainnekuvin ja sanallisella esittelyllä. Käytännön huomioita esitettiin perustuen mobiilikompassin toteutusprojektista saatuihin kokemuksiin. Kompassin käyttökelpoisuutta nykymaailmassa pohdittiin erityisesti siihen vaikuttavien virhe- ja häiriölähteiden kautta. Kompassisuuntaa vertailtiin sen yleisimpään vaihtoehtoon satelliittipaikannukseen.

Tutkitussa kirjallisuudessa esitetyssä suuntalaskennassa magneettikenttää kuvaava vektori pyöritetään gravitaation osoittamalle tasolle ja kalibroidaan laitekohtaisella tekijällä. Tämän jälkeen vaakasuuntainen suuntakulma kertoo magneettisen pohjoissuunnan. Tulos suodatetaan alipäästösuodattimella ja kompensoidaan erannolla. Piirivalmistajan kuvaamassa kalibrointilaskennassa erotetaan laitekohtainen sisäinen häiriötekijä ulkoisesta kentästä pienimmän neliösumman menetelmällä. Insinöörityön tutkimuksissa havaittiin, että kompassieranto on mahdollista selvittää myös automaattisella ohjelmistoratkaisulla. Kaupunkiympäristössä esiintyvien häiriötekijöiden huomattiin vaikeuttavan kompassin käyttöä navigaatiovälineenä. Näiden vuoksi kompassi ei ehkä sovellu nykyisin kovin hyvin pääasialliseksi navigaatiovälineeksi, ja sitä käytetäänkin usein satelliittipaikannuksen tukena integroiduissa järjestelmissä. Moottoriajoneuvojen todettiin synnyttävän erityisiä haasteita.

Työtä voidaan käyttää antamaan kokonaisvaltainen käsitys sekä kompassilaskennasta itsestään että mobiilikompassiin liittyvistä käytännön kysymyksistä. Analyysit käytännön käytettävyydestä ja vertailu yleisimpään vaihtoehtoiseen välineeseen tukevat päätöksentekoa toteutusprojektiin ryhtymisestä. Häiriötekijöiden jatkotutkimus voisi edesauttaa navigaatiojärjestelmien kehitystä.

Avainsanat	kompassi, mobiililaite, anturi, hard iron, eranto, kalibrointi

Author Title Number of Pages Date	Jussi Nikula Implementing a Compass on a Mobile Device 24 pages + 2 appendices 31 May 2018
Degree	Bachelor of Engineering
Degree Programme	Land Surveying
Instructor	Jussi Laari, Senior Lecturer

This bachelor's thesis aimed to provide an understandable overview of the implementation of a compass application by utilizing the sensors on a mobile device. The thesis was to fulfil a void between brief superficial summaries and heavy mathematical presentations about the topic.

Chip manufacturer documents were overviewed and calculations described with illustrations and textual presentation. Additional practical insights were provided based on experiences from an implementation project. The usability of the compass in the modern world was discussed, and compared with satellite positioning.

The orientation and calibration calculations presented in the studied literature rotate the magnetism vector with the direction of gravity and calibrate it with a device specific term, which is distinguished from the external magnetic field using the least squares method. Research revealed that the local magnetic declination can be determined automatically. The thesis established that electromagnetic disturbances in urban environment degrade compass performance and, therefore, the compass might be better for a supporting role in integrated systems. Motorized vehicles were discovered to be challenging use cases.

This thesis can be used to provide a general understanding of compass calculation and related practical matters. The presented analyses of practical usability and comparison with the most common alternative support decision making.

Keywords	compass, mobile device, sensor, hard iron, magnetic declination, calibration

Sisällys

Lyhenteet ja käsitteet

alipäästösuodatus	Elektroniikassa yleisesti käytetty suodatusmenetelmä, joka päästää vain riittävän pienen vaihtelun lävitseen. Kynnysarvoa korkeammat arvot suodatetaan lopputuloksesta.
eranto	Maapallon magneettikentän suunnan ja karttapohjoisen välinen ero. Magneettinen deklinaatio. Erannolla korjataan magneettista pohjoisnapaa osoittava kompassilukema osoittamaan maantieteellistä pohjoisnapaa. Eranto vaihtelee ajallisesti ja paikallisesti.
GNSS	Global Navigation Satellite System. Satelliittipaikannuksen teknologiariippumaton yleisnimi.
hard iron	Laitteeseen vaikuttavan magneettikenttähäiriön laitekohtainen osuus. Laitteen sähköisesti aktiivisten komponenttien synnyttämä magneettikenttä. Laitteen sisäinen häiriötekijä.
IGRF	International Geomagnetic Reference Field. Kansainvälinen geomagneettinen referenssimalli. Maan magneettikenttää kuvaava malli.
permeabilisuus	Fysikaalinen suure, joka kuvaa materiaalin magneettista läpäisevyyttä. Synonyymi permeabiliteetille. Verrattavissa sähkönjohtavuuteen eli konduktiivisuuteen.
projektio	Vektorin vertailulinjan suuntainen osa. Vektorin kohtisuora “heijastus” vertailulinjalla.
sensori	Anturi. Elektroninen havainnointilaite.
skalaariluku	Matriisimatematiikan termi, joka tarkoittaa normaalia reaalilukua, joka ei ole matriisi. Engl. *scalar*.

soft iron	Laitteeseen vaikuttavan magneettikenttähäiriön laitteen ulkopuolisista tekijöistä johtuva osuus. Ympäristössä olevan magnetismin induktiovaikutus laitteen magnetisoitumattomiin, mutta permeabilisiin osiin kuten metallisiin suojakuoriin. Laitteen ulkoinen häiriötekijä. Induktioilmiö on riippuvainen laitteen ominaisuuksista, vaikka kentän lähde onkin laitteen ulkopuolella.
transpoosi	Matriisi, joka saadaan, kun alkuperäisen rivit käännetään sarakkeiksi ja päinvastoin. Transpoosissa matriisi ikään kuin peilautuu lävistäjänsä suhteen. Engl. *transpose*. Transpoosin transpoosi on sama kuin alkuperäinen luku.
WMM	World Magnetic Model. Yhdysvaltain ja Yhdistyneiden kuningaskuntien sotilashallintojen ja maantieteellisten tutkimuslaitoksien yhdessä kehittämä maan magneettikenttää kuvaava malli.
yksikkömatriisi	Matriisialgebrassa skalaarilukujen ykköstä vastaava neutraali arvo. Yksikkömatriisilla kertominen ei muuta arvoa. Matriisi, jonka lävistäjällä on ykkösiä ja muissa paikoissa nollia. Engl. *identity matrix*.

1 Johdanto

Modernit mobiililaitteet sisältävät sensorit eli anturit magnetismille ja kiihtyvyydelle. Näistä voidaan muodostaa kompassi, joka laskee sensorien tuottamista vektoreista maan pinnan suuntaisen kompassisuuntakulman. Asiaa käsittelevät verkosta löytyvät lähteet ovat vaihtelevan tasoisia, sisältäen joko raskasta matemaattista teoriaa ilman kansantajuista yleisesittelyä tai ollen vain kevyitä esittelyitä, joista ei käy ilmi kaikki tarpeellinen tieto älypuhelinkompassin toteuttamiseksi. Tämä insinöörityö pyrkii toimimaan ymmärrettävänä esittelynä asiasta, jonka avulla voidaan muodostaa älypuhelinkompassin toteuttamisesta hyvä kokonaiskuva. Yleiskäsityksen avulla on helpompi sisäistää kompassilaskentaa kuvaavia dokumentteja ja ymmärtää niissä esitettyjen laskentamallien merkitys.

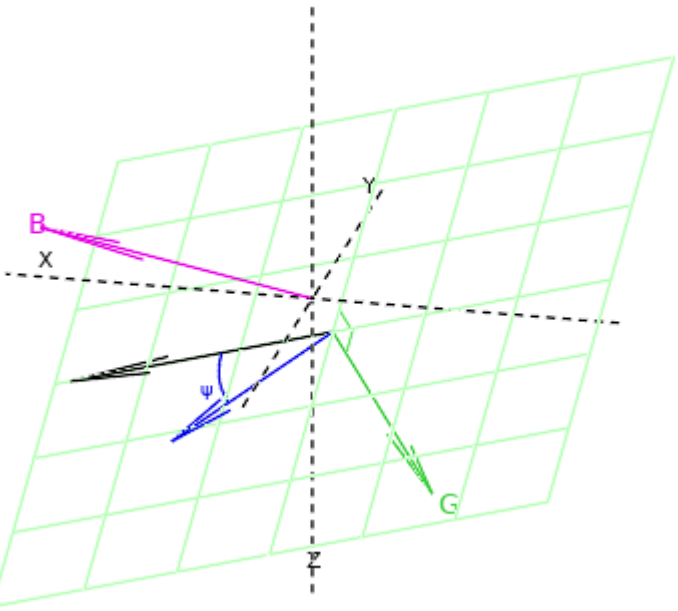
Työn lähtökohtana toimii kirjoittajan työharjoitteluna San Sai Solutions Oy:lle kesällä 2017 tekemä työ, jossa toteutettiin kompassi Android-puhelimeen Embarcadero Delphi -sovelluskehittimellä osana tuotekehitysprojektia. Tämä työ perustui pääasiassa sovellusinsinööri Talat Ozyagcilarin kirjoittamiin ja piirivalmistaja Freescale Semiconductorin julkaisemiin dokumentteihin aiheesta, joissa kuvataan itse kompassisuunnan laskeminen ja toisaalta magneettisensorien kalibrointi (1; 2).

Toimivan mobiilikompassin toteuttaminen vaatii käytännössä itse suunnan laskennan lisäksi myös kalibrointilaskennan. Jokaisella laitemallilla on sille ominainen oma magneettikenttänsä, jonka synnyttää sen sisältämä elektroniikka. Tämän vaikutus magneettisensorin tuottamaan dataan täytyy suodattaa, jotta voidaan mitata ympäristössä vallitsevaa magneettikenttää. (1; 2.)

Tässä työssä esitellään piirivalmistaja Freescale Semiconductorin julkaisemia kompassisuunnan ja kalibroinnin laskentamalleja ja analysoidaan niiden toimintaperiaatteita. Myös kompassin mobiililaitetoteutukseen liittyviä käytännön näkökulmia esitellään. Lisäksi pohditaan kompassin käyttökelpoisuutta käytännön navigointivälineenä nykymaailmassa, ja käsitellään kompassin näyttämään suuntaan vaikuttavia virhe- ja häiriölähteitä. Kompassia vertaillaan sen yleisimpään vaihtoehtoon, satelliittipaikannukseen, käytännön käyttökelpoisuuden näkökulmasta.

2 Käsittely

2.1 Kompassisuunnan laskeminen



Kuva 1. Kompassisuunta gravitaatiotason projektiona.

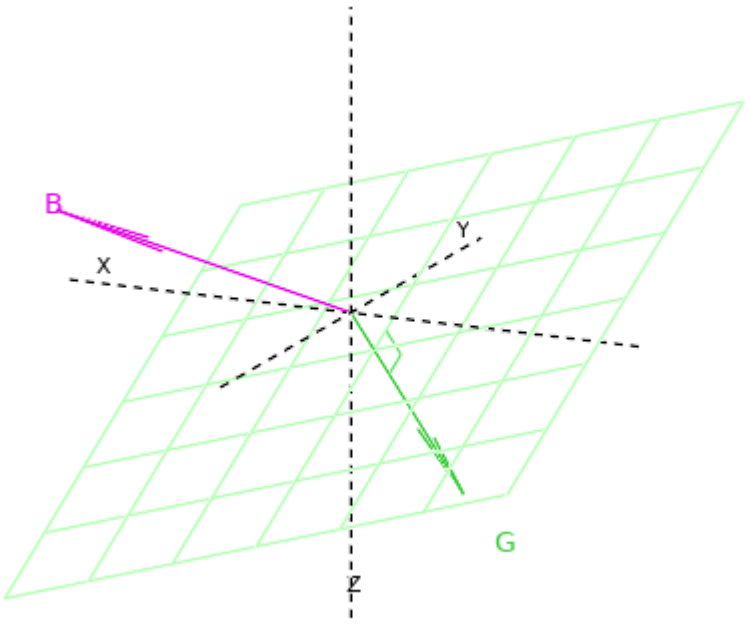
Kuva 1 esittää laskennasta yleiskäsityksen. Se sisältää tärkeimmät lähtötiedot ja laskennan tavoitteen: maan pinnan tason mukainen pohjoissuunnan kertova kulma. Kuva ei täysin vastaa suuntalaskentaa, joka toimii siirtämällä mittaustiedot koordinaatistoakselien tasolle, eikä koordinaatistoa gravitaatiotasolle. Sen tarkoitus onkin pääasiassa havainnollistaa tehtävän tavoitetta eikä niinkään itse laskentaa. Tämän insinöörityön havainnekuvat on laadittu Octave-ohjelmistolla. Octave on yhteensopiva avoimen lähdekoodin vastine kaupalliselle Matlab-ohjelmistolle. Insinöörityön liitteissä 1 ja 2 on annettu tarvittava ohjelmakoodi, jolla havainnekuvat voidaan toistaa kolmiulotteisina esityksinä Matlabissa tai Octavessa. Koska käsiteltävä asia on kolmiulotteinen, kuvat ovat kolmiulotteisina tasoesitystä havainnollisempia. Havainnekuvissa on käytetty vektoriprojektiota orientoimaan vektoreita tason mukaisiksi. Tämä ei vastaa suuntalaskennan matematiikkaa, jossa ei käytetä projektiolaskentaa, vaan kulmarotaatiota. Vektoriprojektiosta poiketen kulmarotaatio säilyttää vektorien pituudet ennallaan.

2.1.1 Lähtötiedot

Kompassisuunnan laskemiseen tarvitaan useita lähtötietoja. Nämä ovat

- kiihtyvyysanturin mittaustiedot
- magnetismianturin mittaustiedot
- paikallinen kompassieranto
- sisäisen häiriövaikutuksen kalibrointitiedot
- ulkoisen induktiovaikutuksen kalibrointitiedot.

Laitteesta riippuen ulkoisen induktiovaikutuksen kalibrointi voidaan sivuuttaa, ja silti saada käytännössä käyttökelpoisia tuloksia (1, s. 2; 2, s. 2). Kuva 2 näyttää suuntalaskennan lähtötilanteen: anturien tuottamat mittausvektorit puhelimen koordinaatistossa.



Kuva 2. Kompassilaskennan tärkeimmät lähtötiedot: gravitaatio- ja magnetismivektorit puhelimen koordinaatistossa.

Tärkeimmät lähtötiedoista ovat laitteen anturien tuottamat kiihtyvyyden ja magnetismin mittausvektorit. Älypuhelimen sisältämät magnetismi- ja kiihtyvyysanturit antavat sensoritiedon vektorimuodossa. Vektori esitetään kolmena lukuarvona, jotka ovat vektorin x,y,z -komponentit. Näistä olisi helppo tehdä se olettamus, että vektorin komponentit olisivat jossain tietyssä merkityksellisessä asennossa, kuten että z-komponentti kertoisi

pystysuuntaisen asennon joko laitteeseen tai maan pintaan nähden. Tällaista oletta-musta ei kuitenkaan voi suoraan tehdä. Vektorikomponenttien suunnat riippuvat laitearkkitehtuurista ja laitteen suunnittelussa tehdyistä päätöksistä (1, s. 3). Tämän vuoksi mittaustiedoista tuleekin varmistaa, missä asennossa antureiden koordinaatisto on. Tämä tapahtuu asettamalla laite tunnettuun asentoon ja lukemalla anturien antamat lukemat (1, s. 4). Koska laskennan tuloksena syntyy suuntakulma koordinaatiston akseliin nähden, laitteen koordinaatiston asentoa laitteeseen nähden voidaan pitää yhtenä tarvittavista lähtötiedoista. Koordinaatiston “kalibrointia” käsitellään jäljempänä laskennan yhteydessä.

Gravitaatiovektori kertoo laitteeseen kohdistuvan kiihtyvyyden suunnan ja suuruuden. Laitteen ollessa levossa tämä osoittaa maan vetovoiman suunnan. Koska gravitaatio havaitaan kiihtyvyyttä mittaavalla anturilla, kompassilaskenta ei voi tuottaa oikeaa suuntaa, mikäli laite on kiihtyvässä liikkeessä tai vapaassa pudotuksessa (1, s. 6). Tässä työssä käytetään kiihtyvyysanturista myös termiä gravitaatioanturi tai gravitaatiosensori, vaikka kyseessä tarkkaan ottaen onkin kiihtyvyyden mittaus. Tämän työn kannalta kiihtyvyysanturia ei käytetä muuhun tarkoitukseen kuin gravitaation eli putoamiskiihtyvyyden mittaukseen. Vetovoimatiedon perusteella saadaan taso, jonka suuntainen magneettikenttää kuvaavan vektorin orientaatiokulma kertoo kompassisuunnan. Gravitaatiosensorin tuottamien vektorien yksikkönä on yleensä kiihtyvyyden yksikkö g. Näin oli ainakin kirjoittajan tekemissä käytännön kokeissa Android Nexus 5 -puhelimella: gravitaatiosensorin z-komponentti osoitti lukemaa n. 1.0 laitteen ollessa levossa pöydällä. Standardi putoamiskiihtyvyys maan pinnalla on noin 1 g (3). Tämä epästandardi yksikkö voidaan merkitä joko isolla G-kirjaimella tai pienellä (3). Se onkin helppo sekoittaa toisaalta galileoon, joka on myös kiihtyvyyden yksikkö, tai lyhenteensä perusteella joko suuruusluokkaetuliitteeseen giga (G) tai massan yksikköön gramma (g). Sensorin mittaustietojen yksikkö ei kuitenkaan ole kompassisovelluksen kannalta merkityksellinen, koska kompassilaskenta perustuu suuntakulmiin ja suunta on sama yksiköstä riippumatta.

Magnetismianturi puolestaan kertoo anturipiirin kohdalla vallitsevan magneettikentän suunnan. Tyypillisesti magnetismianturin tuottama vektoridata on yksiköltään mikroteslоina (μT). Tesla kuvaa magneettivuon tiheyttä (4, s. 12 ja 19). Magnetismianturin tuottama vektori kertoo anturin kohdalla vallitsevan magneettikentän suunnan ja voimakkuuden. Magnetismianturin tietojen yksikkökään ei ole kompassilaskennan kannalta olennainen tieto, koska kentän suunta ei riipu sen voimakkuudesta.

Magneettikenttään vaikuttavat voimakkaimmin laitteen oman elektroniikan ja metallisten komponenttien tuottamat vaikutukset (5, s. 3; 1, s. 9). Nämä voidaan jakaa sisäiseen ja ulkoiseen vaikutukseen (hard iron ja soft iron). Sisäinen vaikutus syntyy laitteen elektroniikan tuottamista sähkömagneettisista kentistä (5, s. 4). Ulkoinen vaikutus kuvaa laitteen ulkopuolisen magneettikentän induktiovaikutusta sen sisältämiin metallikuoriin tai akkuihin (5, s. 5). Magnetismianturin tuottama tieto on kalibroitava näihin tekijöihin nähden, jotta voidaan havainnoida laitteen ulkopuolista magneettikenttää. Kalibrointitiedot annetaan syötteenä suunnan laskennalle (1; 2). Käytännössä voidaan saada käyttökelpoisia tuloksia myös jättämällä ulkoinen häiriötekijä huomiotta (1, s. 2), mikäli laite on suunniteltu siten, että nämä vaikutukset eivät ole erityisen voimakkaita (2, s. 2).

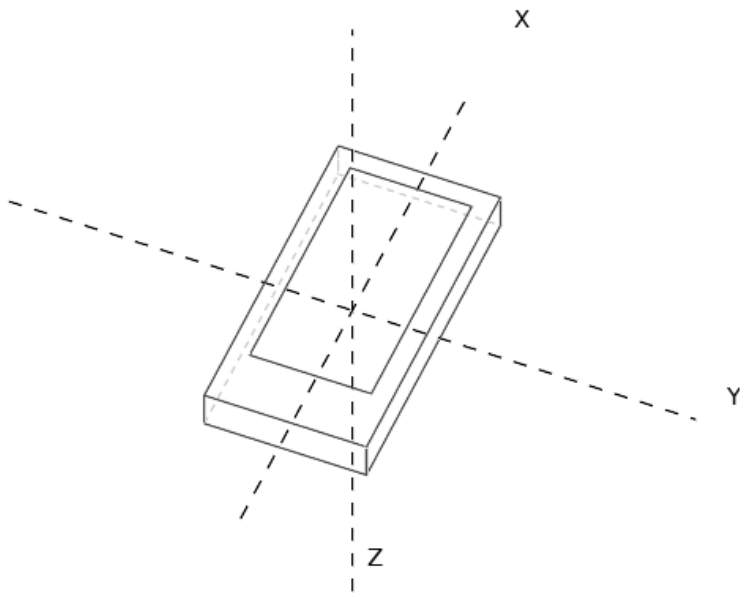
Todellisen pohjoissuunnan selvittämiseksi kalibroidusta lopputuloksesta täytyy myös kompensoida kompassieranto eli magneettinen deklinaatio. Tieto paikallisen erannon suuruudesta on yleensä helppo löytää verkosta. Erannon selvittäminen on myös mahdollista toteuttaa automatisoidusti. Erantoa käsitellään laajemmin kalibroinnin yhteydessä jäljempänä.

2.1.2 Suunnan laskeminen lähtötiedoista

Talat Ozyagcilar esittelee teoksessaan *Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors* ratkaisun kompassisuunnan laskemiseksi mobiililaitteen kiihtyvyys- ja magnetismiantureiden tuottamista mittaustiedoista. Teoksessa esitelty laskenta soveltaa rotaatiomatriisilaskentaa ja trigonometriaa vektoreihin ja sisältää kompensaation laitteen sisäiselle häiriötekijälle.

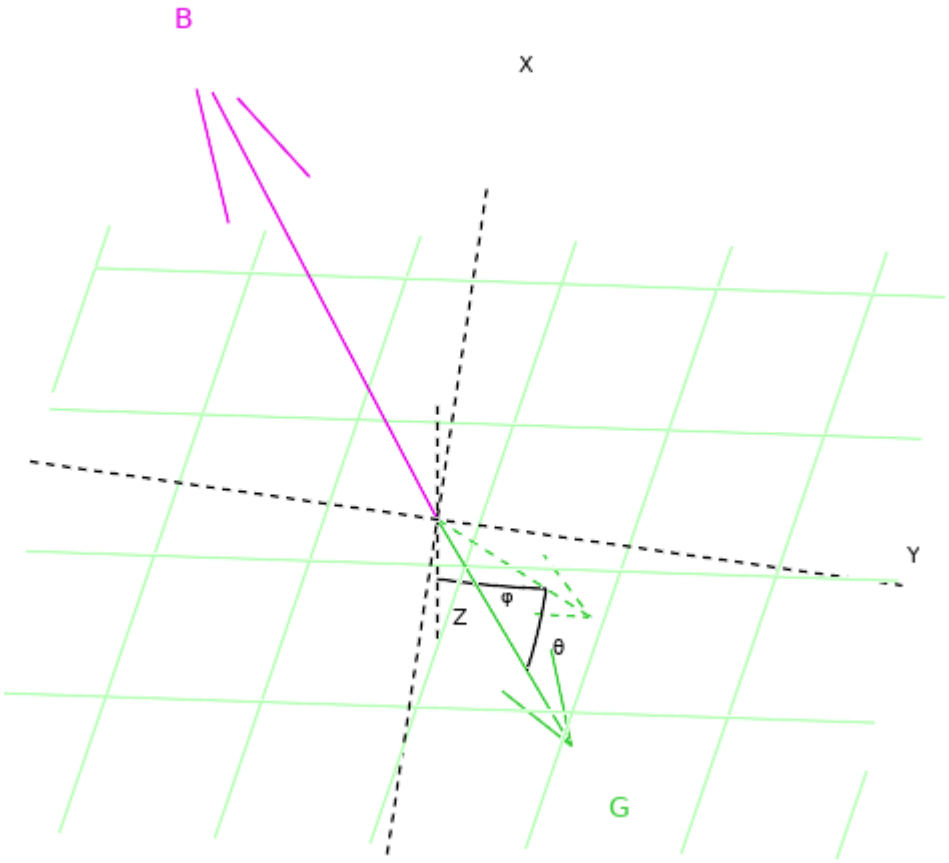
Ozyagcilarin esittämä suunnan laskenta ratkaisee vain gravitaatiovektorin ja magneettikenttävektorin välisen projektio-ongelman ja kalibroinnin vaatiman kompensaation. Se ei ratkaise kysymystä siitä, miten mittausantureiden tuottamat vektorikomponentit asettuvat itse laitteeseen nähden (1, s. 3). Tyypillisesti anturien antamat vektorikomponentit orientoituvat laitteen mukaisesti, ja gravitaatio- ja magneettikenttävektorit ovat keskenään samassa koordinaatistossa (1, s. 3). Jos vektorit eivät olisi samansuuntaisessa koordinaatistossa toisiinsa nähden, kompassisuunnan ratkaiseminen vaatisi näiden keskinäisen orientaatioeron tuntemista ja rotaatiota yhtenevään orientaatioon. Jääkin sovelluskehittäjän tehtäväksi selvittää ja varmistaa vektorikomponenttien ja laitteen keskinäinen orientaatio. Mikäli vektorikomponenttien orientaatio poikkeaa Ozyagcilarin teoksen

sivuilla 3 ja 4 esitetyistä lähtöolettamuksista, tämä voidaan ratkaista esimerkiksi negatoimalla vektorikomponentteja tai järjestelemällä niitä uudelleen. Tämä tulee tehdä ennen edellä esitettyä ratkaisulaskentaa anturien tuottamille vektoreille. Jos komponentit eivät ole ollenkaan puhelinlaitteen suuntaisia, voidaan ne pyörittää rotaatiomatriisilla vastaamaan kompassilaskennan lähtöoletuksia käyttäen testaamalla löydettyjä kulmia. Jos x- ja y-komponentit ovat samassa tasossa laskennan lähtöoletuksiin nähden, voidaan korjaus tehdä myös laskennan lopputuloksena saatuun kulmalukemaan. Laskettu suunta on tällöin edelleen käyttökelpoinen, mutta sen nollapiste saattaa osoittaa muualle kuin pohjoiseen tai kulmalukema kasvaa vastapäivään. Nämä voidaan korjata lisäämällä tai vähentämällä suuntakulmasta suoria kulmia tai negatoimalla kulmalukema. Kuva 3 esittää laskennan lähtöoletuksien mukaista puhelimen koordinaatiston asentoa.



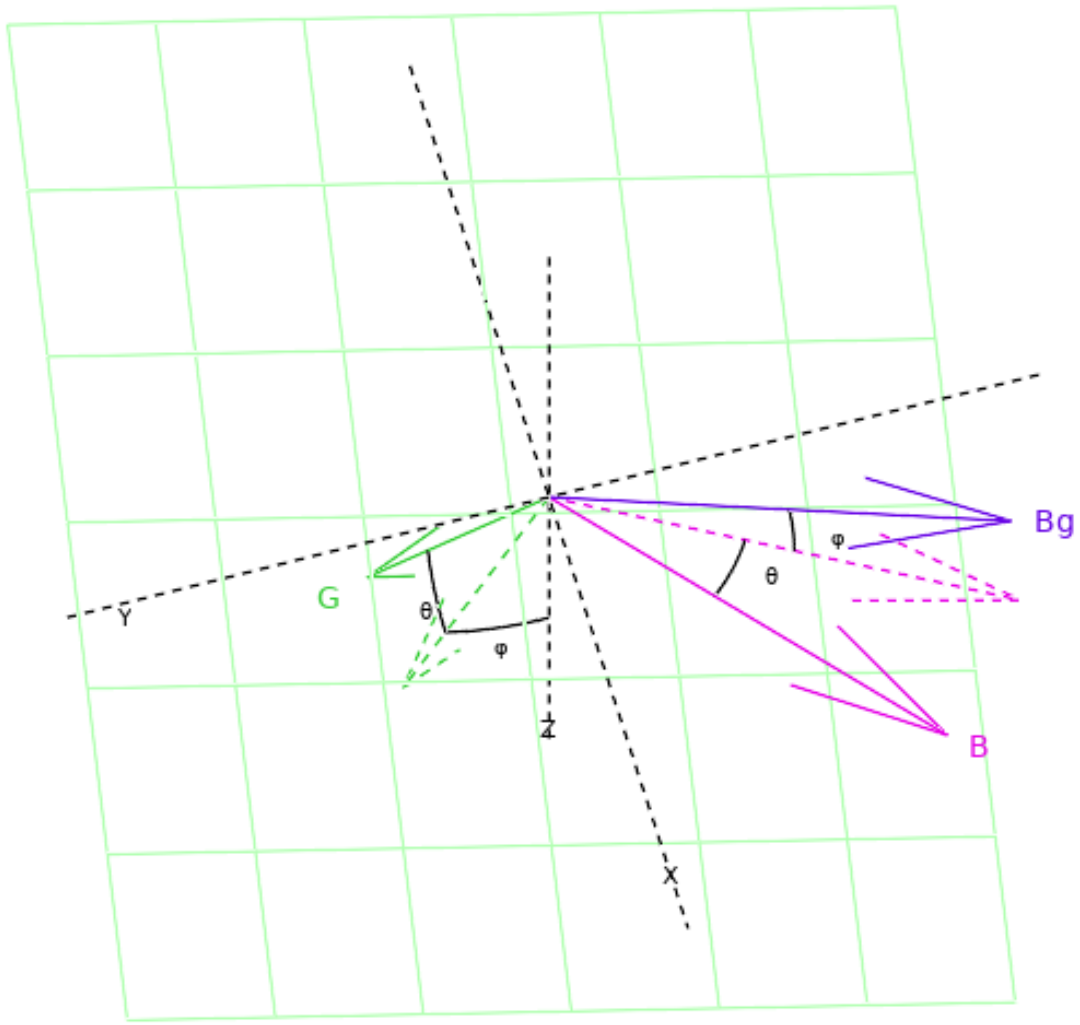
Kuva 3. Laskennan lähtöoletus mobiililaitteen koordinaatiston asennosta. Koordinaatiston akselit kasvavat niiden otsikkokirjainten osoittamiin suuntiin.

Suunnan laskenta on Ozyagcilarin dokumentissa johdettu matemaattisesti sivuilla 5–8. Tämän toteutuksesta on esitetty malliratkaisu sivuilla 15–16. Edellä mainituissa otetaan huomioon vain laitteen sisäisen häiriötekijän kalibrointi. Ulkoisen induktiovaikutuksen kalibrointia käsitellään laajemmin jäljempänä kalibroinnin yhteydessä.



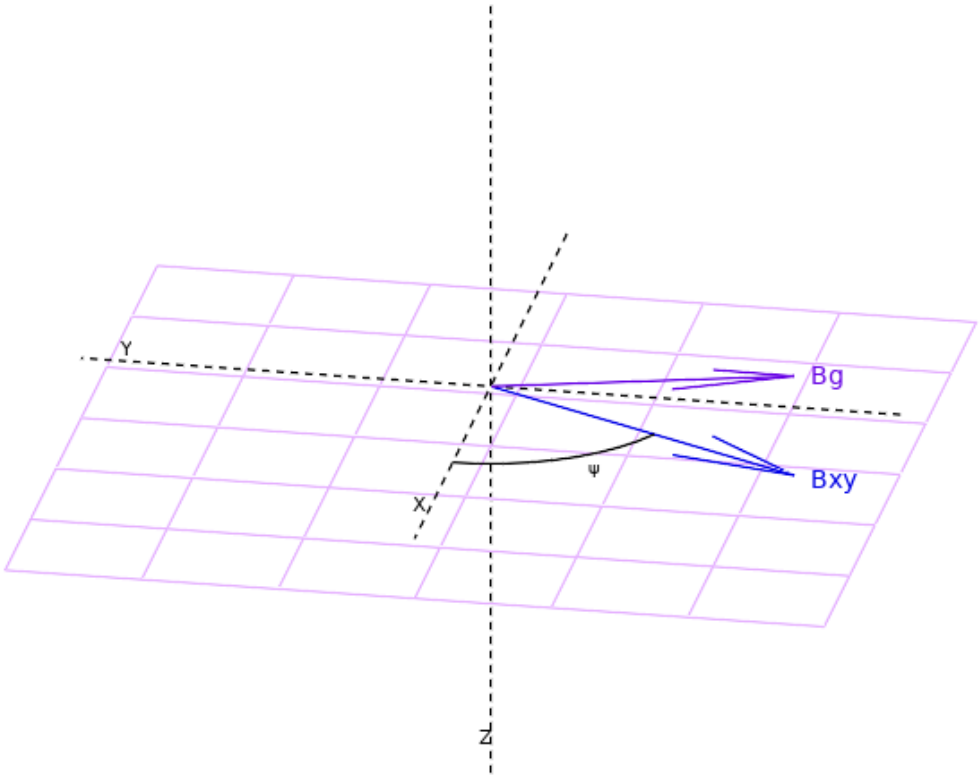
Kuva 4. Kulmat phi φ ja theta θ saadaan gravitaation suunnasta.

Suuntalaskenta alkaa hakemalla gravitaatiovektorin perusteella kaksi suuntakulmaa (φ ja θ), jotka osoittavat vetovoiman suunnan laitteeseen nähden (1, s. 7). Näiden avulla magneettikentän kuvaaja voidaan orientoida maan vetovoiman osoittamalle tasolle. Lisäksi magneettikenttää kuvaavasta vektorista vähennetään kalibrointivektori, jolla sisäinen häiriötekijä poistetaan tuloksesta (1, s. 8). Kuva 4 osoittaa gravitaatiosta saatavien kulmatietojen sijainnin ja merkityksen kallistumana z-akseliin nähden. Kuva 5 puolestaan havainnollistaa, miten saatuja kulmia käytetään orientoimaan magnetismivektori uudelleen. Orientoinnin jälkeen magnetismivektori on sellaisessa asennossa, että koordinaatiston xy-taso vastaa maan pinnan asentoa magnetismiin nähden.



Kuva 5. Magnetismin orientointi gravitaation mukaiseksi.

Maan pinnan tason mukaiseksi pyöritetystä korjatusta magneettikenttävektorista voidaan tämän jälkeen ratkaista sen vaakasuuntainen suuntakulma koordinaatistoon nähden. Koska pyrkimyksenä on ratkaista kompassisuunta, yhtälöt on järjestelty siten, ettei magneettikentän pystysuuntaista kulmaa (δ) tarvita vaakasuuntaisen kulman (ψ) ratkaisemiseksi. Kulmat täytyy myös rajata välille −180 ja 180 astetta, sillä muuten yhtälöillä olisi rajaton määrä ratkaisuja. Koska koordinaatisto on molemmissa mittausvektoreissa laitteen mukaisessa orientaatiossa, kulma ψ riittää kertomaan laitteen suunnan ja magneettisen pohjoissuunnan erotuksen. (1, s. 7–8.) Kuva 6 esittää lopputuloksena saatavaa vaakasuuntaista kulmaa. Kuvassa näkyvä kuvitteellinen taso havainnollistaa sitä, että gravitaation kulmilla orientoinnin jälkeen maan pinta on ikään kuin siirtynyt koordinaatiston tason mukaiseksi.



Kuva 6. Kompassisuunta XY-tasolla, kun kalibroitu magnetismivektori on pyöritetty gravitaation mukaiseen suuntaan. Havainnekuvan taso esittää kuvitteellista maan pinnan tasoa pyörityslaskennan jälkeen. Vektori Bxy esittää gravitaatiokorjatun magnetismivektorin Bg projektiota XY-tasolla vaakasuuntaisen kulman psi ψ eli kompassisuunnan havainnollistamiseksi. Laskenta ei todellisuudessa käytä projektiota. Vaakakulma lasketaan trigonometrisesti vektorista Bg.

2.1.3 Lasketun suunnan alipäästösuodatus

Kalibroitunakin magneettikenttäympäristö sisältää kohinaa eli korkeataajuuksista vaihtelua. Esimerkiksi vaihtosähkövirta tuottaa modulaatiota magneettikenttään (5, s. 7). Myös laitteen sisäinen elektroniikka saattaa synnyttää kohinaa, jota ei voi kalibroimalla saaduilla vakiotermeillä täysin suodattaa. Lisäksi maan magneettikenttä saattaa sisältää lyhytaikaisia vaihteluja (4, s. 10), esim. auringon aktiivisuuden synnyttämää kohinaa (4, s. 15–16). Kohinailmiön vuoksi laskettu kompassisuunta täytyy suodattaa alipäästösuodattimella.

Ozyagcilar esittelee suunnan laskentaa kuvaavassa työssään myös tähän tehtävään soveltuvan suodatusalgoritmin. Suodatus on käytännössä yksinkertainen laskutoimitus. Se perustuu liukuvaan keskiarvoon, jossa jokaisen näytteen erotuksesta aiempiin arvoihin

nähden lisätään tulokseen vain tietty halutun näytemäärän mukainen osa. Lisäksi lopputuloksena syntyvä suuntakulma normalisoidaan −180 ja 180 asteen välille. (1, s. 16–17.) Näytemäärän valinta on tasapainottelua responsiivisuuden ja tarkkuuden välillä. Suurempi näytemäärä parantaa tarkkuutta, mutta hidastaa reagointia suunnan muutoksiin. Reaktionopeutta voidaan parantaa myös tihentämällä päivitystaajuutta, mutta tälläkin on oma hintansa kasvavana prosessointitehon tarpeena.

Ozyagcilarin esittelemää suodatinkoodia tulkitessa on hyvä huomata, että siinä esitetty vakioarvo ANGLE_LPF (32768 / N) on suodatuksen säätöarvo vain siltä osin kuin se sisältää näytemäärän lukeman N. Lukuarvo 32768 vakiossa on vain kompensaatio bittien siirrolle 15 bittiä oikealle. $32768 = 2^{15}$. Kertomalla ennalta lasketulla arvolla 32768/N ja siirtämällä bittejä korvataan jakolasku kokonaislukukertolaskulla. Tämä operaatio on mitä todennäköisimmin vain ohjelmointiteknistä optimointia, jolla ei ole merkitystä suodatuksen kannalta. Tekstissä mainittu termi *modulo arithmetic* viitannee kulmien erotuksen normalisointiin välille −180 ja 180 astetta ennen jakolaskua, eikä käytettyyn kokonaislukukertolaskumenetelmään. Käytännössä voidaankin hyvin käyttää normaalia jakolaskuoperaatiota. Tällöin suodatuslaskennan rivi

```
tmpAngle = (Int32)iLPPsi + ((ANGLE_LPF * tmpAngle) >> 15);
```

korvautuu jakolaskulla, esim.

```
tmpAngle = iLPPsi + tmpAngle / N;
```

Kirjoittajan oman kokemuksen mukaan jakolaskun käyttämisestä ei ollut mitään havaittavissa olevaa haittaa lopputuloksen tarkkuuteen. Suorituskyky saattaa kärsiä, mutta luettavuus paranee. Käytetty rajapinta antoi mittausnäytteet liukulukuina, jolloin oli luonnollista käyttää liukulukuaritmetiikkaa, jonka yhteydessä bittien siirto-operaatio ei ole toimiva optimointimenetelmä.

2.1.4 Magneettipohjoisesta karttapohjoiseen

Laskennan lopputuloksena saatavasta magneettipohjoisen suuntakulmasta päästään todelliseen karttapohjoisen suuntaan kompensoimalla kompassieranto eli magneettinen deklinaatio yksinkertaisella summalaskulla. Eranto on yleisen käytännön mukaan positiivinen silloin, kun magneettipohjoinen on karttapohjoisesta itään (4, s.13). Ozyagcilarin mukaan suuntalaskennan lopputuloksena saatava kulma psi ψ kasvaa myötäpäivään

pohjoisesta alkaen (1, s. 3). Tämä on kuitenkin riippuvainen puhelimen koordinaatiston asennosta. Onkin hyvä varmistaa testaamalla, kumpaan suuntaan laskettu kulma-arvo kasvaa ja missä ilmansuunnassa sen nollapiste on. Myötäpäivään kasvavaan suuntakulmaan tehdään erantokorjaus vähentämällä eranto suuntakulmasta. Deklinaatiokorjaus voidaan tehdä joko ennen alipäästösuodatusta tai sen jälkeen. Koska suodatus perustuu kulmien keskinäisiin erotuksiin, deklinaatiokorjaus ei olennaisesti vaikuta sen toimintaan.

2.2 Kalibrointi

Kompassisovelluksen kalibrointi koostuu useista tekijöistä. Kalibrointitekijät ovat

- laitteen sisäinen häiriötekijä (hard iron)
- laitteen ulkoinen induktiovaikutus (soft iron)
- kompassieranto.

Kompassieranto ja laitteen itsensä tuottama sähkömagneettinen kenttä (hard iron) on kalibroitava, jotta voidaan saada käyttökelpoisia kompassituloksia. Laitteen ulkoinen induktiovaikutus (soft iron) voidaan sen sijaan joissain tapauksissa jättää huomiotta (1, s. 2; 2, s. 2).

2.2.1 Laitteen sisäinen häiriötekijä (hard iron)

Magneettikenttää lukevan anturin mittauksiin vaikuttavat voimakkaimmin laitteen oman elektroniikan synnyttämät sähkömagneettiset kentät (5, s. 3; 1, s. 9). Nämä vaikutukset on kalibroitava, jotta anturilla voidaan lukea laitteen ulkopuolista magneettikenttää. Talat Ozyagcilar esittelee teoksessaan *Calibrating an eCompass in the Presence of Hard- and Soft-Iron Interference* kalibrointimenetelmän, jolla laite voidaan kalibroida ympäristöstä riippumatta. Tämä on mahdollista, koska laitteen ominaiskenttä liikkuu laitteen mukana, kun taas ulkoinen kenttä muuttuu laitteeseen nähden sen asennon muuttuessa. Tämä mahdollistaa kalibroinnin laitetta kääntelemällä. Kalibrointiohjelmisto ottaa näytteitä laitteeseen vaikuttavasta magneettikentästä. Näiden perusteella voidaan magneettikentän havainnointitiedoista laskennallisesti erottaa laitteen asennon myötä muuttuva osa ennallaan pysyvästä. Se osuus kentästä, joka pysyy (laitteeseen nähden) aina samana asennosta riippumatta, on laitteen oma ominaiskenttä. (1, s. 6.)

2.2.2 Laitteen ulkoinen häiriötekijä (soft iron)

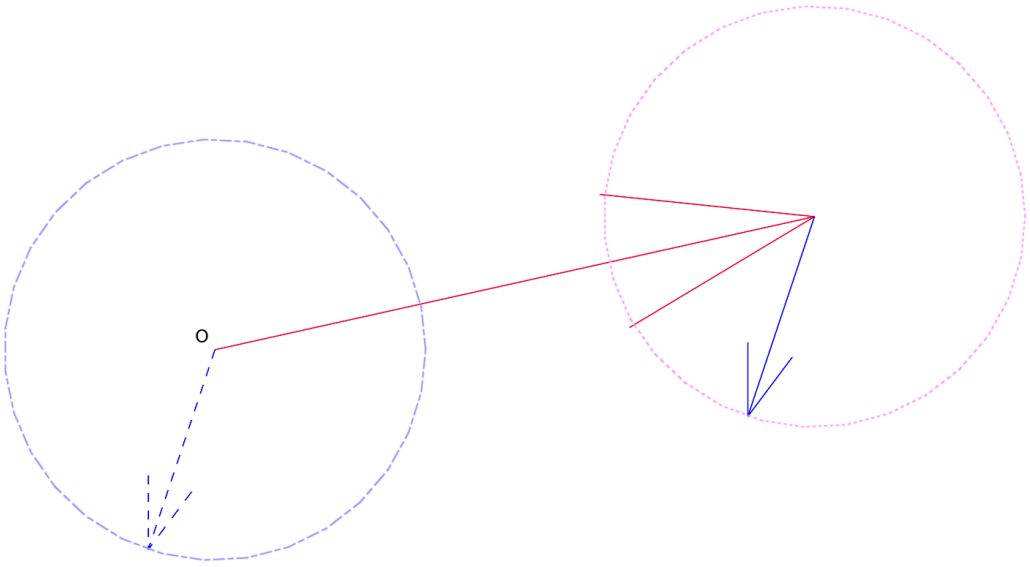
Laitteen itsensä tuottaman sähkömagnetismin lisäksi häiriötä aiheuttaa myös laitteen ulkopuolisen magneettikentän vaikutus sen sähköisesti passiivisiin mutta sähköä johtaviin osiin, kuten metallisiin suojakuoriin. Ulkoinen magneettikenttä indusoituu laitteen metalliosiin ja akkuihin. Tätä vaikutusta kutsutaan termillä *soft iron*. Sen kalibrointi on monimutkaisempi tehtävä kuin sisäisen tekijän, koska ilmiö ei ole lineaarinen. (5, s. 5; 6, s. 13.) Induktiovaikutus on riippuvainen käytetyn materiaalin permeabilisuudesta (5, s. 8; 7, s. 662–663). Permeabilisuus on materiaalin fysikaalinen ominaisuus, joka kuvaa sen magneettista läpäisevyyttä (8, s. 102, 104, 157, 191).

Koska induktiovaikutus ei ole lineaarinen ilmiö, sen kalibrointiin tarvitaan useita tekijöitä. Yhdellä 3x3-matriisilla voidaan mallintaa magneettikentän ja sen synnyttämän induktion keskinäinen suhde (W_{Soft}), ja lisäksi tarvitaan vektorit esittämään anturin komponenttien suorakulmaisuuden puutteet ($W_{NonOrthog}$) ja komponenttien mahdolliset erisuuruisuudet (W_{Gain}). Nämä voidaan yhdistää matriisitulolla yhdeksi 3x3-matriisiksi (W), jota voidaan käyttää ulkoisen vaikutuksen kalibroinnin kuvaajana. (2, s. 3–5.)

2.2.3 Kalibrointilaskenta

Ozyagcilar kuvaa kalibrointia käsittelevässä työssään laskentamallin, jolla voidaan löytää laitteen sisäistä häiriötekijää kuvaava vektori (2, s. 12–16). Kalibrointilaskennassa etsitään sellainen kalibrointivektori, joka sovittaa kalibroimattoman näyteaineiston origokeskiseksi palloksi. Kalibroinnin lähtötietoina käytetään useita näytemittauksia laitteen magneettikentästä. Laadukkaan ja käyttökelpoisen kalibrointituloksen aikaansaamiseksi mittausnäytteiden tulee edustaa vaihtelevia laitteen asentoja. Muutoin ulkoinen ja sisäinen kenttä eivät erotu toisistaan, ja kalibroinnilla löydetään vain magnetometrin mittauksien satunnainen kohinavaihtelu. (2, s. 15.) Gravitaation mittausta tarvitaan kalibrointiin vain siltä osin, että sitä käyttäen voidaan tunnistaa laitteen asennon muutokset heterogeenisen näyteaineiston keräämiseksi. Näyteaineistosta muodostetaan yhtälöryhmä, joka voidaan ratkaista pienimmän neliösumman menetelmällä. Näin saadaan tulokseksi magnetismin kompensaatiovektorin x-, y- ja z-komponentit, sekä magneettikentän voimakkuus. Ratkaistava yhtälöryhmä muodostetaan mittausnäytteiden ja kalibrointivektorin summasta.

Kuva 7 on yksinkertaistettu tasoesitys kalibroinnin geometrisesta merkityksestä. Laite ja käyttäjä ovat aina kuvan origossa (O). Kuvasta voidaan nähdä, miten maan magneettikenttää voimakkaampi hard iron -tekijä vääristää mittausnäytteet osoittamaan aina samaan sektoriin. Todellisuudessa ilmiö on luonnollisesti kolmiulotteinen.



Kuva 7. Sisäisen tekijän (hard iron) kalibrointilaskennan yksinkertaistettu graafinen esitys. Punainen vektorinuoli kuvaa laitteen sisäistä häiriötekijää. Sininen vektorinuoli kuvaa maan magneettikenttää. Sinipunainen ympyrä esittää kalibroimattomien mittaustuloksien joukkoa ja sininen ympyrä samaa joukkoa kalibroinnin jälkeen.

Pienimmän neliösumman menetelmälle tyypillisesti kalibrointituloksen laatu paranee näytemäärän kasvaessa, ja näytteitä tarvitaan vähintään tuntemattomien tekijöiden verran. Koska tuntemattomia tekijöitä oli neljä, kalibrointiyhtälön ratkaisemiseksi tarvitaan vähintään tämän verran näytteitä. Laadukkaan tuloksen saamiseksi on kuitenkin suositeltavaa ottaa huomattavasti tätä useampia näytteitä, koska ylimääräytyminen tarkentaa ratkaisua.

Kalibrointinäytteiden ottamisen ohjelmointia ei ole kuvattu Ozyagcilarin teoksessa. Näytteenotto voidaan toteuttaa esimerkiksi lukemalla ajastettuna magnetismi- ja kiihtyvyysanturien tietoja, vertailemalla kiihtyvyysanturin lukemaa edellisellä kerralla saatuun ja tallentamalla magnetismin näytearvo vain, jos kiihtyvyysanturin mittaustieto poikkesi riittävästi aiemmasta tilanteesta. Näin voidaan varmistaa, että kalibrointilaskennan näyteaineisto on riittävän heterogeenista hyvän tuloksen saamiseksi. Kun näytteitä on saatu riittävästi, voidaan siirtyä laskemaan kalibrointikorjaus aineistosta.

Ozyagcilarin kalibrointidokumentissa esitellään vain sisäisen häiriötekijän (hard iron) ratkaiseva kalibrointilaskenta esimerkkikoodina. Tämä on kalibroinnin yksinkertaistettu versio, jossa ei oteta induktiovaikutusta huomioon lainkaan. Yksinkertaistetun kalibrointiratkaisun lisäksi dokumentissa esitellään ulkoisen induktiovaikutuksen (soft iron) huomioon ottava lisäys Ozyagcilarin suunnan ratkaisua kuvaavassa dokumentissa (1) esiteltyyn ohjelmakoodiin. Myös induktiovaikutuksen kalibroinnin matematiikkaa käsitellään, mutta valmista vastausta ei anneta. Valmis induktiovaikutuksen kalibrointiratkaisu on kuitenkin Ozyagcilarin mukaan saatavilla julkaisijalta veloituksetta tuotteisiin, jotka käyttävät Freescalen magnetometrejä (2, s. 17). Induktiovaikutuksen kalibroinnissa haetaan sisäisen tekijän lisäksi soft iron -ilmiötä kuvaava kalibrointikerroinmatriisi. Induktiovaikutus ei ole sisäisen tekijän tapaan lineaarinen, joten sen kalibrointimatriisi on mittausnäytteille kerroin eikä summan termi.

Tämän insinöörityön kirjoitushetkellä Freescale Semiconductorin www-sivusto (http://www.freescale.com) ohjaa yhteydenotot yrityksen nimeltä NXP Semiconductors N.V. sivustolle (http://www.nxp.com). Tämän perusteella voidaan olettaa, että yritykset ovat fuusioituneet toisiinsa. Kirjoittaja ei ole ottanut kumpaankaan yritykseen yhteyttä, joten dokumentissa mainitun tarjouksen voimassaolo ei ole tiedossa. Tämän työn lähdeluettelossa listattu osoite on kuitenkin voimassa ainakin toistaiseksi. Freescale Semiconductorin vuonna 2015 julkaisemat pdf-dokumentit sisältävät jo NXP:n logon, ja ne ovat edelleen esillä yli kaksi vuotta julkaisuhetkensä jälkeen. Tämä antaa toivoa siitä, että NXP ei ole aikeissa poistaa niitä julkisuudesta. Dokumentit ovat korkealaatuisimmat kirjoittajan verkkohaulla löytämät mobiililaitekompassin toteuttamista kuvaavat julkaisut, joten on oletettavaa, että niille on edelleen kysyntää.

Kalibrointilaskenta vaatii käytännössä jonkinlaisen matriisimatematiikan työkalun joko itse toteutettuna tai ulkopuolisena kirjastona. Matriisialgebran operaatiot on aina mahdollista toteuttaa auki kirjoitettuna ilman matriiseja, mutta tämä olisi käytännössä työläämpää ja virhealttiimpaa kuin matriisiolion toteuttaminen. Kalibrointilaskenta käyttää matriisikertolaskua, transpoosia ja käänteismatriisia (2, s. 12–16). Jotta matriisioliosta tulisi yleiskäyttökelpoinen työkalu, siihen kannattaa toteuttaa myös yhteen- ja vähennyslaskentaoperaatiot ja yksikkömatriisin (*identity matrix*) alustus. Matriisioperaatioista haastavin on käänteismatriisin laskeminen. Käänteismatriisiongelman ratkaiseminen on mahdollista useilla erilaisilla menetelmillä (9, s. 290–291, 301, 973; 10). Käänteismatriisi on matriisien vastine skalaarilukujen käänteisluvulle (9, s. 290; 10).

Ozyagcilarin esittämän kalibrointilaskennan ymmärtämisen helpottamiseksi esitetään tässä vielä joitakin huomioita:

Vektorin transpoosin matriisitulo vektorin itsensä kanssa antaa tuloksena skalaarin, joka on alkuperäisen vektorin komponenttien neliöiden summa. Tämä voidaan tarkistaa esim. *Matlab*-ohjelmalla tai sen ilmaiseksi saatavilla olevilla vastineilla *Freemat* ja *Octave*:

```
>> r = [1;2;3]
r =
   1
   2
   3

>> transpose(r)*r
ans =  14

>> 1^2+2^2+3^2
ans =  14

>> norm(r)^2
ans =  14.000

>> dot(r,r)
ans =  14
```

Esimerkkikoodi 1. Vektorin transpoosin ja vektorin itsensä matriisitulo.

Tämä pätee vain vektoreihin, joita esittävässä matriisissa on yksi sarake, ja vain tässä järjestyksessä. Muutoin matriisitulo tuottaa tulokseksi matriisin, jolla on useampi ulottuvuus. Transpoosin matriisitulo vastaa myös vektorin pistetuloa itsensä kanssa (9, s. 716). Jos vektorit annetaan ryhmänä yhdessä matriisissa, ryhmän transpoosin tulo itse vektoriryhmän kanssa tuottaa matriisin, jonka lävistäjällä on ryhmän kunkin jäsenvektorin komponenttien neliöiden summa:

```
>> r = [1 5; 2 10; 3 25]
r =
   1    5
   2   10
   3   25

>> transpose(r)*r
ans =
    14   100
   100   750

>> 1^2+2^2+3^2
ans =  14

>> 5^2+10^2+25^2
ans =  750
```

Esimerkkikoodi 2. Vektoriryhmän transpoosin matriisitulo ryhmän itsensä kanssa.

Nämä esimerkit selittävät sitä, miksi laskentakaavoissa on käytetty paljon tuloa X^TX, ja myös yhtälön 21 kalibrointidokumentin sivulla 12. Ozyagcilar esittää myös laajan matemaattisen johdon, jolla lopullisiin kalibrointiyhtälöihin on päädytty.

Matriisimatematiikassa ei ole jakolaskua. Jos tulon tekijä siirretään yhtälön puolelta toiselle, tämä ratkaistaan kertomalla käänteismatriisilla. Käänteismatriisilla kertominen vastaa siis matriisialgebrassa skalaarialgebran jakolaskua. Kalibrointilaskennassa oleva käänteismatriisi tuleekin lopulliseen kaavaan siitä, että sillä siirretään yhtälöstä matriisitekijä yhtäsuuruusmerkin toiselle puolelle: jos $X^TX\beta = X^TY$, silloin $\beta = (X^TX)^{-1}X^TY$. (9, s. 812; 10.)

Kalibrointilaskennan johdoksen lopputuloksena saatava yhtälö noudattaa myös Metropolian maanmittauksen koulutusohjelman matematiikan kursseilta tuttua pienimmän neliösumman ratkaisukaavaa yhtälöryhmälle: $(A^T A)^{-1} A^T L$. Yhtälöryhmän tekijöiden kertoimet annetaan toisella matriisilla (A), ja vakiotekijät toisella (L).

2.2.4 Kompassieranto

Kalibroidulla kompassilaskennalla saadaan selville maan magneettikentän paikallinen suunta ja voimakkuus. Maan magneettikentän suunnan ja maantieteellisen pohjoisnavan suunnan välistä eroa kutsutaan kompassierannoksi eli magneettiseksi deklinaatioksi (4, s. 13). Maan magneettikenttä vaihtelee paikallisesti ja myös ajallisesti (4, s. 12, 15–16, 92–105), joten eranto ei ole kiinteä vakiotermi.

Magneettikentän suunnasta saadaan karttapohjoisen suunta, kun se kompensoidaan paikallisella deklinaatiolla. Deklinaatio voidaan kompensoida laskettuun magneettipohjoiseen joko manuaalisesti säädettävällä paikallisen deklinaation säätöarvolla tai automaattisesti. Paikallisen deklinaation arvo manuaalista säätöä varten voidaan selvittää useista eri verkkolähteistä. Esimerkiksi alla mainittu Yhdysvaltain kansallinen ympäristöinformaatiokeskus tarjoaa myös tällaisen palvelun.

Yhdysvaltain valtionhallinnon kansallinen ympäristöinformaatiokeskus *National Centers for Environmental Information* (NCEI) esittelee verkkosivuillaan ohjelmiston, jolla paikallinen eranto voidaan automaattisesti laskea. Ohjelmisto sisältää tietokannan, jossa on tiedot magneettikentän paikallisesta vaihtelusta. Tietokanta perustuu Yhdysvaltain ja Yh-

distyneiden kuningaskuntien puolustushallintojen ja maantieteellisten tutkimuskeskuksien yhteistyönä kehittämään World Magnetic Model (WMM) -malliin. Mallia päivitetään viiden vuoden välein ja sillä voidaan myös ennustaa lähitulevaisuuden tilannetta. (11) Kun NCEI:n automaattinen erantolaskenta yhdistetään Ozyagcilarin esittämään ympäristöstä riippumattomaan kalibrointiin, voidaan laatia kompassisovellus, joka ei vaadi lainkaan säätöarvoja käyttäjältä ja toimii lähes kaikkialla.

Edellä mainitun WMM:n lisäksi käytössä on myös muita maan magneettikenttää kuvaavia malleja. Kansainvälinen geomagneettinen referenssimalli IGRF on ollut käytössä jo vuosikymmeniä (4, s. 35). Myös IGRF-mallilla voidaan tehdä ennusteita maan magnetismin lähitulevaisuuden tilanteesta. Ilmatieteen laitoksen Heikki Nevanlinnan mukaan korkeintaan 5 vuoden päähän ekstrapoloiminen on mahdollista lineaarisesti (4, s. 35). Ei liene sattumaa, että Nevanlinnan mainitsema aikaväli on saman mittainen kuin WMM-mallin päivitysjakso.

Maan magneettikenttä vaihtelee sekä ajallisesti että paikallisesti. Ajalliset vaihtelut voidaan jakaa lyhytjaksoiseen transienttimuutokseen ja pitkäjaksoiseen sekulaarimuutokseen (4, s. 15). Lisäksi esim. rautamalmiesiintymät aiheuttavat paikallisia poikkeamia magneettikenttään (4, s. 16). Edellä esitetyt mallit kuvaavat hyvin pitkäjaksoisia ajallisia muutoksia. Myös paikallisesti esiintyvät stabiilit poikkeamat voidaan kartoittaa ja sisällyttää malliin. Lyhytjaksoinen ajallinen vaihtelu on kuitenkin vaikeasti ennustettavaa, koska se aiheutuu pääasiallisesti auringon aktiivisuudesta ja on luonteeltaan epäsäännöllistä (4, s. 141–154, s. 165). Tästä seuraa se, että deklinaation lyhytjaksoiset muutokset jäävät väistämättä kompassitulokseen mukaan ja saattavat aiheuttaa virhettä. Kompassisovellukselle valittua päivitystaajuutta lyhytjaksoisemmat jaksolliset häiriöt tasoittunevat ainakin osittain alipäästösuodatuksessa, mutta päivitystaajuutta pidempijaksoiset vaihtelut jäävät kompassin tulokseen virheeksi.

2.2.5 Muut häiriötekijät

Nykyaikaisessa urbaanissa ympäristössä on runsaasti sähköjohtoja. Sähkövirta synnyttää aina ympärilleen magneettikentän (4, s. 12, 108, 112–113; 6, s. 9–10, 12–13; 7, s. 605–628; 8, s. 190). Ilmiön havaitsi ensimmäisenä Hans Christian Örsted vuonna 1820 (4, s. 108, 112; 12, s. 117–120). Örstedin havainto oli merkittävä sähkömagnetismin tutkimuksen virstanpylväs (4, s. 169). Tällaiset ympäristön kentät aiheuttavat kompassiin

häiriöitä (4, s. 112–113), jotka vähentävät kompassin käyttökelpoisuutta navigaatiovälineenä kaupunkiympäristössä. Näiden kalibrointi vaatisi kartoitustietoa vallitsevasta sähkömagnetismista, eli tietoa kaikkien sähköjohtojen sijainneista ja niiden kuljettaman sähkövirran voimakkuudesta ja suunnasta. Tämä on käytännössä mahdotonta, joten tätä häiriötekijää ei voi kalibroimalla poistaa. Ilmiö vaikuttaa myös perinteiseen kompassiin. Kirjoittajan älypuhelinkompassin parissa tekemän testaustyön yhteydessä tämä kävi ilmi siten, että etäisyys testauspaikan vieritse kulkevaan sähköistettyyn rautatiehen vaikutti käytännössä suoraan verrannollisesti kompassisovelluksen näyttämään suuntaan. Ilmiön voi havaita viemällä kompassin lähes minkä tahansa sähköisesti aktiivisen laitteen lähelle. Myös Nevanlinna viittaa teoksessaan kaupungistumisen Helsingin magneettiselle observatoriolle aiheuttamiin ongelmiin (4, s. 177).

Myös infrastruktuurin metalliset massat saattavat toimia rautamalmiesiintymien tavoin, ja vaikuttaa joko maan magneettikenttään tai vierellä kulkevan sähköjohdon tuottamaan kenttään. Tässä on kyse samasta induktioilmiöstä kuin soft iron -häiriössä. Kenttä indusoituu permeabiliseen materiaaliin.

2.3 Käytäntöä

Aiemmin kompassi oli erittäin hyödyllinen ja tarpeellinen navigaatioväline erityisesti meriliikenteessä. Kun Kolumbus purjehti Amerikkaan, kompassi oli tärkeässä asemassa (4, s. 94). Sittemmin, toisaalta satelliittipaikannuksen yleisen saatavuuden paranemisen ja toisaalta kaupunkiympäristön häiriötekijöiden kasvamisen myötä, sen merkitys ja tarpeellisuus ovat vähentyneet (4, s. 95).

Kalibroinnin yhteydessä esitettyjen häiriötekijöiden vuoksi kompassi on navigaatiovälineenä käyttökelpoisin maaseudulla, merellä ja ilmaliikenteessä. Kirjoittajan oman kokemuksen mukaan sen yleisin vaihtoehto satelliittipaikannus toimii huonosti vesillä. Tämä havaittiin Metropolia AMK:n kurssin yhteydessä luotaamalla suoritetuissa lammen tilavuusmittauksissa, jolloin monitieheijastus lammen pinnan ja pohjan kautta aiheutti ongelmia satelliittimittaukseen. Kyseisen lammen savesta samea vesi tosin myös luultavasti korosti ongelmaa. Puhtaampivetisessä vesistössä monitieheijastusvaikutus ei välttämättä ole yhtä voimakas. Myös tiheän puuston alueella satelliittipaikannus on ongelmallista (13, s. 280, 529). Rakennusten sisällä satelliittipaikannus on käytännössä hyödytöntä, koska rakennusten seinät ja katto estävät tehokkaasti satelliittisignaalin kulun

(13, s. 280, 529; 14, s. 310–311). Lisäksi suunnan määritys satelliittipaikannussijaintien perusteella vaatii sijainnin muuttumista satelliittipaikannuksen erottelutarkkuuden verran, jotta voidaan tunnistaa paikan muuttuminen ja laskea suunta kahden eri sijainnin kesken. Paikallaan pysyttäessä satelliittipaikannuksella voidaan määritellä suunta vain erityisillä antennijärjestelyillä, joilla voidaan tunnistaa satelliitin suunta havaintolaitteeseen nähden (14, s. 9–10, 12, 270–271). Käytännössä tällaisia antenneja tuskin on mobiililaitteissa. Android Nexus 5:n GNSS-piirin tuottama suunta ei päivittynytkään paikallaan ollessa. Kompassisuunta toimii myös näissä tilanteissa, joten se voikin tällöin olla hyödyllinen väline. Toisaalta magneettisesti aktiivisessa ympäristössä älypuhelimen satelliittipaikannuspiirin tuottama suuntakulma on parempi suunta-arvio, sillä se ei ole niin altis erilaisille magneettikentän häiriöille.

Sekä kompassi että sen yleisin vaihtoehto satelliittipaikannus kärsivät häiriöistä tietyissä tilanteissa. Koska kompassi ei sähkömagneettisten kenttien yleisyyden vuoksi ole kovin käyttökelpoinen kaupunkiympäristössä, se ei ehkä sovellu kovin hyvin pääasialliseksi navigaatiovälineeksi ainakaan tällaisissa ympäristöissä. Parhaimmillaan se lieneekin apuvälineenä kompensoimaan satelliittipaikannukseen perustuvan navigaatiojärjestelmän puutteita.

Koska satelliittipaikannukseen perustuva suunnan määritys vaatii käytännössä tunnistettavan sijaintieron, jalan kuljettaessa sen tuottama suunta päivittyy melko harvakseltaan. Älypuhelinten GNSS-ratkaisujen eli satelliittipaikannuksen erottelukyky on useiden metrien luokkaa. Käytännössä Android Nexus 5:llä tämä oli n. 5–10m kirjoittajan testikokemusten perusteella. Tämän vuoksi kävellen liikkuessa GNSS-suunta päivittyy harvakseltaan. Kompassi onkin käyttömukavuuden kannalta miellyttävämpi menetelmä kävellen, sillä sen suunta-arviota voidaan päivittää vapaavalintaisella taajuudella. Korkeamman päivitystaajuuden ansiosta kompassisuunta soveltuu myös karttanäkymän asemointiin ympäristöön nähden GNSS-suuntaa paremmin. Kompassisuunnan mukaan näytöllä ympäristön mukaiseen asentoon asettuva karttanäkymä on miellyttävä apuväline auttamaan kartan ja ympäröivän todellisen maailman välisen suhteen hahmottamisessa. Moottoriajoneuvoilla liikuttaessa satelliittisuunta päivittyy nopeammin suurempien nopeuksien ansiosta, jolloin sen käytettävyys on parempi.

Suunnanmääritys mobiililaitteella kuljettaessa moottoriajoneuvolla onkin erityisen haastava tehtävä. Moottoriajoneuvon tyypillisesti metallinen katto ja seinät haittaavat satelliittipaikannuksen signaalia (14, s. 310). Tämä johtuu joko Faradayn häkkinä tunnetusta

ilmiöstä (15, s. 441; 16, s. 259) tai sitten yksinkertaisesta katvevaikutuksesta rakennusten ja maamassojen tapaan. Todennäköisesti kyse on suurimmaksi osaksi jälkimmäisestä, koska puhelinten muut toiminnot toimivat käytännössä autossakin aivan hyvin, eli auton kori ei ainakaan kokonaan pysäytä niiden tiedonsiirtoon käyttämää radiosignaalia. Toisaalta moottoriajoneuvo on myös sähkömagneettisesti aktiivinen laite, mikä puolestaan synnyttää kompassiin häiriötä. Auton runko toimii yleensä osana sen virtapiiriä (17, s. 511), jolloin se tuottaa myös oman sähkömagneettisen kenttänsä. Ozyagcilarin mukaan Maan magneettikenttä on voimakkuudeltaan hyvin heikko verrattuna mobiililaitteiden tuottamaan kenttään (1, s. 9; 5, s. 3). Saman voidaan hyvin olettaa pätevän autoonkin. Tämän vuoksi kompassi todennäköisesti toimii autossa hyvin huonosti, ja virheellinenkin satelliittipaikannussuunta on luultavasti parempi arvio kulkusuunnasta.

Insinöörityön yhteydessä suoritetuissa kokeissa Google Play -sovelluskaupasta Samsung XCover 3 -puhelimeen (Android) ladatuilla kompassisovelluksilla (*Compass 360 Pro* ja *Compass*) havaittiin, että henkilöauton sisällä kompassisuunta muuttui riippuen sijainnista auton sisätilassa. Yleisesti ottaen kompassisuunta poikkesi samassa paikassa ilman autoa tehtyyn vertailumittaukseen nähden n. 0–30 astetta. Erityisesti etäisyys auton oviin asennettuihin kaiuttimiin vaikutti havaittavasti kompassisuuntaan. Keskelle auton sisätilaa keskikonsolin päälle paikalleen asetetun puhelimen osoittama kompassisuunta pysyi ennallaan, kun autolla ajettiin n. 0,5–1m suoraan eteen- tai taaksepäin. Kun puhelimen sijaintia muutettiin saman etäisyyden verran samaan suuntaan niin, että auto pysyi paikallaan ja puhelin siirtyi auton keskeltä kojelaudan vierelle, kompassisuunnassa havaittiin n. 30 asteen muutos. Moottorin sammuttaminen tai käynnistäminen ei vaikuttanut havaittavasti auton sisätilassa vallitsevaan magneettikenttään. Kokeissa käytetty ajoneuvo oli bensiinikäyttöinen Ford Mondeo -henkilöauto vuosimallia 1999. Kokeet suoritettiin 16.4.2018. Sama koe tehtiin kahdessa eri paikassa: pysäköintialueella katujen Vanha Maantie ja Maantienpelto risteyksessä Espoossa, ja opiskelijaasuntolan pysäköintipaikalla osoitteessa Kilonrinne 10, Espoo. Molemmissa kokeissa havaittiin sama ilmiö: kojelaudan vieressä vallitseva kenttä poikkesi merkittävästi sisätilan keskellä olevasta.

Mahdollisia lähestymistapoja ajoneuvossa tapahtuvaan suunnanmääritykseen voisivat olla keskiarvolaskenta GNSS-suunnista, esimerkiksi toisaalla tässä työssä kuvatulla liukuvan keskiarvon menetelmällä, tai kompassisovelluksen kalibroiminen ajoneuvon sähkökentän mukaisesti. Kompassin kalibroiminen ajoneuvon mukaan tarkoittaisi ajoneuvon tuottaman kentän mukaan ottamista kalibrointilaskentaan. Tämä vaatisi kiinteää asentoa

mobiililaitteelle ajoneuvossa, ja on ongelmallista, koska ajoneuvon asennon kääntely kalibrointiprosessia varten on huomattavasti mobiililaitteen kädessä pyörittelyä vaikeampaa. Kalibrointi tuottaa parhaat tulokset, kun näytteet on otettu mahdollisimman heterogeenisista asennoista (2, s. 15), ja ajoneuvon asentoa ei helposti voi maan pinnan tasosta muuttaa. Riittävän kaltevalla tasolla voitaisiin ehkä päästä käytännössä käyttökelpoisiin tuloksiin turvautumatta nosturiin. Ajoneuvon mobiililaitetta huomattavasti suurempimassainen metallirunko todennäköisesti myös voimistaa soft iron -induktioilmiötä, mikä edelleen vaikeuttaa kalibrointia.

Käytännössä paras ratkaisu lieneekin ajoneuvoon integroidun navigointijärjestelmän käyttäminen aina kun se on mahdollista. Tällaiseen integroituun järjestelmään olisi mahdollista lisätä myös magnetismi- ja gravitaatioanturit älypuhelinten tapaan, jolloin sen magnetismianturin sijainti voitaisiin Ozyagcilarin mobiililaitteiden suunnitteluun esittämien ohjeiden (*Layout Recommendations for PCBs Using a Magnetometer Sensor*, tämän insinöörityön lähdeviite 5) avulla valita parhaaksi mahdolliseksi, ja toimiva kompassisuunnan havainnointi voitaisiin näin mahdollistaa. Ajoneuvon oman integroidun järjestelmän suunnittelussa voidaan ottaa huomion sen rungon vaikutus, ja satelliittisignaalia lukeva antenni sijoittaa rungon ulkopuolelle. Tällöin sen tuottama paikannustieto on rungon sisällä olevaan mobiililaitteeseen verrattuna tarkempaa, ja tätä kautta myös suuntatiedon laatu paranee. Jos ajoneuvon järjestelmä mahdollistaa sen tuottamien tietojen lukemisen siihen kytketyllä laitteella, päästäneen parempiin tuloksiin kuin mobiililaitteella yksinään. Ajoneuvonavigaatiojärjestelmissä käytetäänkin jo edellä kuvatun kaltaisia erilaisista lähteistä tietoa yhdisteleviä menetelmiä (13, s. 491–522), ja magnetismianturit ovat mukana näissä yhtenä tietolähteenä (13, s. 509, 514). Magnetismin ja satelliittipaikannuksen lisäksi suunnan muutoksia voidaan havaita myös auton pyörien suhteellista nopeutta mittaamalla (13, s. 509, 514).

3 Yhteenveto

Tässä työssä esiteltiin kompassin mobiililaitesovelluksen toteuttamista, sen vaatimia laskentamalleja ja analysoitiin näiden toimintaperiaatteita. Myös kompassin soveltuvuutta navigointivälineeksi nykymaailmassa pohdittiin. Työn tavoitteena oli hyvän yleiskuvan muodostaminen kompassilaskennasta mobiililaitteessa. Työ kokosi yhteen kompassilas-

kennasta olemassa olevia lähteitä ja esitteli niistä yhteenvedon alkuperäisiä lähteitä selkeämmin. Asiasta annettu kokonaiskuva ja yleiskäsitys antavat hyvän lähtökohdan laskennan ymmärtämiseen, helpottaen näin kompassisovelluksen toteuttamista.

Käytännön käyttökelpoisuudesta esitetyt huomiot ja arviot mahdollistavat kompassitoteutuksen kustannuksien ja siitä saatujen hyötyjen vertailun. Yleiskäsityksen ja käytännön käyttökelpoisuudesta muodostetun kuvan avulla lukija voi helpommin arvioida, onko kompassi tarkoituksenmukainen väline hänen käyttötilanteessaan. Työtä voi käyttää yleiskuvan muodostamiseen aiheesta ennen käytännön toteuttamiseen ryhtymistä. Se voi toimia myös päätöksenteon tukena päätettäessä kompassitoteutuksen tekemisestä tai tekemättä jättämisestä.

Sekä tämän insinöörityön tutkimusten että sen taustalla olevien käytännön kokemuksien yhteydessä havaittiin sähkömagneettisten kenttien yleisyys modernissa kaupunkiympäristössä. Tämä ilmiö hankaloittaa merkittävästi kompassin käyttökelpoisuutta navigaatiovälineenä. Kompassi onkin navigaatiovälineenä käyttökelpoisin alueilla, joilla tällaisia häiriölähteitä on vähän. Kaupunkiympäristössä kompassiin virhettä tuottavat häiriöt tekevät siitä ongelmallisen vaihtoehdon pääasialliseksi suunnanmäärityksen menetelmäksi. Kaupunkiympäristössäkin kompassi on silti hyödyllinen apuvälineenä kompensoimaan muiden menetelmien puutteita. Kuten aiemmin esitettiin, sitä onkin käytetty tällaiseen tehtävään ajoneuvojen navigaatiojärjestelmissä.

Navigointijärjestelmien kehittämisen kannalta edellä mainittujen kaupunkiympäristön sähkömagneettisten häiriökenttien jatkotutkimukselle voisi olla tarvetta. Näiden kenttien vaikutus kompassiin on selvä, ja vaikutuksen mekanismi on pitkään ja hyvin tunnettu sähkövirran ja magneettikentän välinen yhteys. Näiden kenttien vaikutus muihin navigointimenetelmiin sen sijaan ei ehkä ole yhtä hyvin tutkittu aihe. Sähkömagneettisten kenttien esiintymistiheydestä ja mahdollisista esiintymien lainalaisuuksista saatu tieto voisi edesauttaa navigointijärjestelmien kehitystä. Myös sitä voisi tutkia, voidaanko navigointilaitteisiin häiriöitä aiheuttavien paikallisten kenttien sijainteja kartoittaa niin, että syntynyttä kartoitustietoa voitaisiin käyttää tarkentamaan navigointijärjestelmiä. Sähköjohtojen sijainteja kartoitetaan jo nyt kunnallistekniikan suunnittelua ja toteutusta varten. On toistaiseksi epäselvää, voisiko tätä työtä hyödyntää laajemminkin. Jos kartoitustiedon perusteella tiedetään, että jonkin kadun alla kulkee sähköjohto, joka synnyttää tietyn suuruisen sähkömagneettisen kentän, voidaan tällaisessa paikassa jättää kompassisuun-

nan tieto käyttämättä, kun tiedetään sen epäluotettavuus. Toisaalta, jos tunnetaan sähköjohdon tarkka sijainti, johdossa kulkevan virran suunta ja suuruus, sekä itse johtimen ja välissä olevan maaperän ominaisuudet, voidaan sen synnyttämän kentän vaikutus ehkä kalibroida ja laskennallisesti saada toimiva kompassisuunta myös voimakkaan häiriökentän alueella. Tässä työssä esitelty kalibrointilaskenta osoittaa, että suunta voidaan laskennallisesti selvittää, vaikka ympäristössä vallitseva sähkömagneettinen kenttä olisi suuruudeltaan huomattavasti maan magneettikenttää voimakkaampi.

Varsinaisten häiriökenttälähteiden lisäksi kompassin käyttökelpoisuuden kannalta voisi olla myös hyödyllistä tutkia infrastruktuurin metallisten massojen vaikutusta. Vaikuttavatko kunnallistekniikan johtoverkoston metalliset rakenteet kompassiin kuten malmiesiintymät? Onko tämä vaikutus suuruudeltaan merkittävä? Jos kadun alla kulkeva sähköjohto menee jätevesikaivon kansiasetelman vieritse, miten tämä vaikuttaa johdon synnyttämään sähkömagneettiseen kenttään?

Osan suuntalaskennan tehtävästä voisi luultavasti toteuttaa myös vektoriprojektiolla. Suuntakulma maan pinnan tasolla voitaisiin laskea myös projisoimalla kalibraatiokorjattu magneettikenttävektori ja koordinaatiston akseli gravitaatiotasolle ja laskemalla näiden projektiovektoreiden välinen kulma. Tämän insinöörityön havainnekuvien kolmiulotteisten kulmakaarien ohjelmoinnissa kävi hyvin ilmi vektorimuunnoksien tehokkuus ja yksinkertaisuus kolmiulotteisessa ympäristössä. Kulmiin perustuva rotaatio on haastavaa kolmiulotteisessa ympäristössä, sillä tällöin voi syntyä kuolleita kulmia, jos pyöritys kulma kerrallaan johtaa tilanteeseen, jossa suunnat yhtenevät toistensa kanssa (14, s. 35). Oikeiden pyörityskulmien ja järjestyksen valinta on myös toisinaan vaikeaa. Vektori-matriisi-algebra ei tosin ole yhtä helposti sisäistettävää kuin rotaatiokulmiin perustuvat muunnokset ja vaatii koulutusta. Se on kuitenkin tietoteknisessä ympäristössä hyödyllistä, koska vektorioperaatiot ovat tietokoneelle tyypillisesti trigonometrisia funktioita vähemmän tehoa vaativia (10). Vaatisi tosin mittauksia ja jatkotutkimuksia, jotta voitaisiin varmasti tietää, olisiko vektoriprojektion käyttäminen juuri tässä tapauksessa Ozyagcilarin esittämää ohjelmakoodia ja laskentaa tehokkaampaa. Ozyagcilarin valitsemaa rotaatiokulmiin perustuvaa lähestymistapaa puoltaa myös kulmien neutraalius vektorisuureen pituuteen nähden. Kulmarotaatio säilyttää vektorin suuruuden ennallaan, toisin kuin projektio.

Tämän työn tekemisessä oli suureksi hyödyksi maanmittauksen koulutusohjelman matematiikan kurssien lisäksi Jaakko Pitkäsen vetämä Metropolian pelikoulutuslinjan Pelimatematiikka 1 -kurssi, joka käsitteli matriisialgebran perusteita ja vektorimuunnoksia. Ilman sitä Ozyagcilarin esittämän matriiseihin ja vektoreihin pohjautuvan matematiikan syvällisempi ymmärtäminen olisi ollut huomattavasti haastavampaa. Kurssia voikin suositella vapaaehtoisena lisäkurssina matematiikasta kiinnostuneille maanmittauksen tutkintoa täydentämään.

Lähteet

1 Ozyagcilar, Talat. 2015. Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors (Application Note 4248). Verkkoaineisto. Freescale Semiconductor, Inc. <https://cache.freescale.com/files/sensors/doc/app_note/AN4248.pdf>. 11/2015. Luettu 28.2.2018.

2 Ozyagcilar, Talat. 2015. Calibrating an eCompass in the Presence of Hard- and Soft-Iron Interference (Application Note 4246). Verkkoaineisto. Freescale Semiconductor, Inc. <https://cache.freescale.com/files/sensors/doc/app_note/AN4246.pdf>. 11/2015. Luettu 28.2.2018.

3 G-voima. 2017. Verkkoaineisto. Wikipedia. <https://fi.wikipedia.org/wiki/G-voima>. Päivitetty 26.10.2017. Luettu 28.2.2018.

4 Nevanlinna, Heikki. 2009. Geomagnetismin ABC-kirja. Verkkoaineisto. Ilmatieteen laitos. <https://helda.helsinki.fi/bitstream/handle/10138/1204/2009nro1.pdf?sequence=1> <http://hdl.handle.net/10138/1204>. 9.4.2009. Luettu 7.3.2018.

5 Ozyagcilar, Talat. 2015. Layout Recommendations for PCBs Using a Magnetometer Sensor (Application Note 4247). Verkkoaineisto. Freescale Semiconductor, Inc. <https://cache.freescale.com/files/sensors/doc/app_note/AN4247.pdf>. 11/2015. Luettu 28.2.2018.

6 Sihvola, Ari & Lindell, Ismo. 2004. Sähkömagneettinen kenttäteoria: 2, Dynaamiset kentät. Helsinki: Otatieto.

7 Benson, Harris. 1996. University physics. Hoboken, NJ: John Wiley & Sons, Inc.

8 Lindell, Ismo & Sihvola, Ari. 2002. Sähkömagneettinen kenttäteoria: 1, Staattiset kentät. Helsinki: Otatieto.

9 Lopez, Robert J. 2001. Advanced engineering mathematics. Boston: Addison-Wesley.

10 Pitkänen, Jaakko. 2017. Pelimatematiikka 1. Kurssiluennot. Metropolia Ammattikorkeakoulu. 23.8.2017–11.10.2017.

11 The World Magnetic Model. Verkkoaineisto. NOAA National Centers for Environmental Information (NCEI). <https://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml>. Luettu 7.3.2018.

12 Lindell, Ismo. 2009. Sähkön pitkä historia. Helsinki: Otatieto Helsinki University Press.

13 Kaplan, Elliott D. & Hegarty, Christopher J. 2006. Understanding GPS – Principles and Applications. Boston: Artech House, Inc.

14 Groves, Paul D. 2013. Principles of GNSS, inertial and multisensor integrated navigation systems. Boston: Artech House.

15 Benenson, W. et al. 2002. Handbook of Physics. New York: Springer-Verlag New York, Inc.

16 Hautala, Mikko & Peltonen, Hannu. 2007. Insinöörin (AMK) fysiikka, osa I. Lahden Teho-Opetus Oy.

17 Automotive electrics, automotive electronics. 2007. Robert Bosch GmbH. Hoboken, N.J: Chichester: John Wiley & Sons, Ltd.

Liite 1. Havainnekuvien lähdekoodi

Tämän työn suuntalaskentaa esittävät havainnekuvat ovat parhaimmillaan kolmiulotteisina esityksinä. Jotta lukija voi halutessaan katsella havainnekuvia kolmiulotteisina, tässä annetaan kuvien tuottamiseksi Matlab- tai Octave-ympäristössä tarvittava lähdekoodi. Liitteessä 1 listattu kuvien piirtokoodi tarvitsee toimiakseen myös liitteessä 2 esitetyt funktiot omissa lähdekooditiedostoissaan. Kuvien tuottamiseksi tarvitaan joko kaupallinen Matlab-ohjelma tai sen avoimen lähdekoodin vastine Octave. Havainnekuvat on laadittu Octave-4.2.2:lla Windows 7-käyttöjärjestelmässä. Muissa käyttöjärjestelmissä tai ohjelman versioissa erityisesti fonttien ulkoasu saattaa vaihdella. Ohjelmakoodi vaatii toimiakseen Matlabissa ohjelman version 2017 tai uudemman.

```
% Script for printing illustrations with Octave or Matlab
% for bachelor's thesis work
% "Implementing a Compass on a Mobile Device"
% Author: Jussi Nikula (2018)

% Dependencies:

% R[3,3] = rotateX(rads)
% R[3,3] = rotateY(rads)
% R[3,3] = rotateZ(rads)
% For generating 3 by 3 rotation matrices for all axes.

% vectorarc()
% For drawing angle arcs to demonstrate angles between 3-dimensional vectors.



% angles for rotating the gravity vector and plane:
% a is for rotating around X axis, b for rotating around Y axis.
a = 20 * pi/180; % 20 degrees in radians
b = 25 * pi/180; % 25 degrees in radians

% Magnetic field vector (column 2) and its start point (column 1) so it's easy
% to move it around if that makes things look better.
% This is drawn to the illustration as is, and the north angle arc and vector
% are computed from this and gravity.
% Note: these are both vectors: the first is the vector from origin to the
% start point and the latter is the vector starting from the first's end.
% This is not the same thing as a pair of start and end points!
B = [0 -12;0 15;0 1];


% Fixed grid with 10 units spacing, depicting Earth surface.
% These are plot3 compatible input, not standard vector matrices.
% All of these must have the same dimensions.
% The gravity plane is intentionally slightly lower than the vectors projected
% onto it so it gets drawn below all the other stuff and our figure looks
% better.
flat_grid_x = [-20 -20 -20 -20 -20 -20 -20 -20 -40/3 -20/3 0 20/3 40/3 20; ...
     20 20 20 20 20 20 20 -20 -40/3 -20/3 0 20/3 40/3 20];
flat_grid_y = [-20 -40/3 -20/3 0 20/3 40/3 20 -20 -20 -20 -20 -20 -20 -20; ...
```

```
     -20 -40/3 -20/3 0 20/3 40/3 20 20 20 20 20 20 20 20];
flat_grid_z = [-3.1 -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 ...
     -3.1 -3.1 -3.1; -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 -3.1 ...
     -3.1 -3.1 -3.1 -3.1];

% Vector pointing to gravity direction
% Since the original surface grid plane had a constant z coordinate,
% This must be aligned with the z-axis to be the surface's normal.
% This time we use the standard vector ordering to make handling easier.
% Note: these are both vectors: the first column is the vector from origin to
% the start point and the latter is the vector starting from the first point.
% This is not the same as a pair of start and end points!
flat_gravity = [0 0;0 0;-3.1 -15];



% Combine the rotational transformations together.
gravity_rotation = rotateY(b)*rotateX(a);

% Rotate all grid coordinates with angles a and b.
[dim_rows,dim_cols] = size(flat_grid_x);
for i = 1:dim_rows
     for j = 1:dim_cols
          % Rotate as a vector so the standard vector rotation matrix system
          % with matrix multiplication works correctly
          xyz = gravity_rotation*[flat_grid_x(i,j);flat_grid_y(i,j); ...
               flat_grid_z(i,j)];

          % Set rotated vector components back to plot3 compatible array form.
          grid_x(i,j) = xyz(1);
          grid_y(i,j) = xyz(2);
          grid_z(i,j) = xyz(3);
     end
end

% Rotate the gravity vectors with the same rotations as the surface plane.
gravity = gravity_rotation * flat_gravity;



% --------------------------------------------------------------------------
% Figure 1: the "big picture" with projections of the magnetic vector on
%             the surface plane
% --------------------------------------------------------------------------

figure(1);
clf;
hold on;

% Set labels for Matlab/Octave coordinate X, Y & Z axes
%    ( mostly for debugging purposes )
xlabel('X');
ylabel('Y');
zlabel('Z');

% Plot the rotated Earth's surface grid first so it usually gets below the
% others in handle order.
% Use a very dim color so the grid doesn't make things too difficult to see.
gridhandle=plot3(grid_x, grid_y, grid_z, 'color',[0.7 1.0 0.7], ...
     'linewidth',1, 'linestyle','-');

% Draw axes representing the device's coordinate system
```

```
plot3([-20;20],[0;0],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[-20;20],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[0;0],[-20;20], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');

% Label the device coordinate axes.
% Octave starts the view with z coord growing upwards, and the camera is at
% a positive z position, so it seemed natural to plot the gravity into
% negative Z, even though in the phone coordinate system Z grows towards the
% bottom of the device. For this reason, the Matlab/Octave coordinate system
% axes directions don't match the phone's axes, though they are otherwise
% aligned with each other.
% The phone's X axis is the Matlab/Octave Y axis, phone's Y is Matlab's X
% and the Z axis is flipped.
textpos = [18;1;1];
H=text(textpos(1), textpos(2), textpos(3), 'Y');
set(H, 'fontsize', 11);
textpos = [0.5;18;1];
H=text(textpos(1), textpos(2), textpos(3), 'X');
set(H, 'fontsize', 11);
textpos = [0.5;0.5;-18];
H=text(textpos(1), textpos(2), textpos(3), 'Z');
set(H, 'fontsize', 11);


% Pick up the rotated coordinates from the rotated gravity vector matrix and
% draw vector arrow with them.
quiver3(gravity(1,1),gravity(2,1), gravity(3,1), gravity(1,2), ...
     gravity(2,2), gravity(3,2), 0, 'color',[0.2 0.8 0.2]);

% Draw magnetic field vector
quiver3(B(1,1),B(2,1), B(3,1), B(1,2), B(2,2), B(3,2), 0, ...
     'color',[0.9 0.0 0.9]);

% Calculate projection of B on the plane of gravity.
% The projection on the plane is the original + the negation of the projection
% on the normal. This does indeed make sense if you think it over graphically:
% the projection on the normal is the same distance and opposite direction
% which you take from the original vector to the plane.
% https://www.opengl.org/discussion_boards/showthread.php/159739-Projection-
of-a-3d-vector-on-a-plane
% is otherwise correct but he forgot to divide v . s with s . s

% Normal of the plane of gravity is the gravity vector: it points the
% orientation of the plane. Its start point may be in different orientation.
gravnormal = gravity(:,2) - gravity(:,1);

% Right angle symbol between gravity vector and gravity plane
vectorarc([grid_x(2,1)-grid_x(1,1); grid_y(2,1)-grid_y(1,1); ...
     grid_z(2,1)-grid_z(1,1)], gravnormal, gravity(:,1), 2, false, true, ...
     false, 10, [0.4 0.9 0.4]);

% Project B onto the normal of gravity:
% ( v . s / s . s ) s    ( dot product is done by v' * s in Matlab )
Bnormal(:,2) = (B(:,2)'*gravnormal) / (gravnormal' * gravnormal) * gravnormal;
Bnormal(:,1) = (B(:,1)'*gravnormal) / (gravnormal' * gravnormal) * gravnormal;

% Projection on the plane is original + negation of the projection on the nor-
mal:
Bp = B - Bnormal;

% We also need to project the phone axis onto the gravity plane
% Using Y axis here since this has all been done in cartesian coordinate sys-
tem.
axvector = [0;20;0];
axnormal = (axvector'*gravnormal) / (gravnormal' * gravnormal) * gravnormal;
```

```
axisp(:,2) = axvector - axnormal;


% Start the projected magnetic north vector from the start point of the
% gravity vector to illustrate that the angle is along the gravity plane:
Bp(:,1) = gravity(:,1);

% Start the axis projection vector from gravity start too
axisp(:,1) = gravity(:,1);

% Draw the projected magnetic north vector
quiver3(Bp(1,1),Bp(2,1), Bp(3,1), Bp(1,2), Bp(2,2), Bp(3,2), 0, ...
     'color',[0.0 0.0 0.9]);

% Draw the axis projection
quiver3(axisp(1,1),axisp(2,1),axisp(3,1),axisp(1,2),axisp(2,2), axisp(3,2),
...
     0, 'color',[0.0 0.0 0.0]);

% Draw angle arc between projected axis and magnetic north
vectorarc(axisp(:,2), Bp(:,2), Bp(:,1), 8, '\psi', true, false, 10, ...
     [0.0 0.0 0.9]);

% Label the gravity and magnetic field vectors
textpos = gravity(:,1)+gravity(:, 2)+gravity(:, 2)*(1/norm(gravity(:, 2)));
H=text(textpos(1), textpos(2), textpos(3), 'G', 'color', [0.2 0.8 0.2]);
set(H, 'fontsize', 14);

textpos = B(:,1)+B(:, 2)+B(:, 2)*(1/norm(B(:, 2)));
H=text(textpos(1), textpos(2), textpos(3), 'B', 'color', [0.9 0.0 0.9]);
set(H, 'fontsize', 14);

% set white background color
set(gcf,'color','w');

axis off;
axis equal;
hold off;




% ------------------------------------------------------------------------
%  Figure 2: just gravity vector and plane
%               + magnetic vector + phone coord axes.
% ------------------------------------------------------------------------


figure(2);
clf;
hold on;
axis off;
axis equal;

% Set labels for Matlab/Octave coordinate X, Y & Z axes
% ( mostly for debugging purposes )
xlabel('X');
ylabel('Y');
zlabel('Z');

% Plot the rotated Earth's surface grid first so it usually gets below the
% others in handle order.
gridhandle=plot3(grid_x, grid_y, grid_z, 'color',[0.7 1.0 0.7], ...
     'linewidth',1, 'linestyle','-');
```

```
% Draw axes representing the device's coordinate system
plot3([-20;20],[0;0],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[-20;20],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[0;0],[-20;20], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');

% Label the device coordinate axes.
textpos = [18;1;1];
H=text(textpos(1), textpos(2), textpos(3), 'Y');
set(H, 'fontsize', 11);
textpos = [0.5;18;1];
H=text(textpos(1), textpos(2), textpos(3), 'X');
set(H, 'fontsize', 11);
textpos = [0.5;0.5;-18];
H=text(textpos(1), textpos(2), textpos(3), 'Z');
set(H, 'fontsize', 11);


% Draw the gravity vector. Start from origin this time.
quiver3(0,0,0, gravity(1,2), gravity(2,2), gravity(3,2), 0, ...
     'color',[0.2 0.8 0.2]);

% Right angle symbol between gravity vector and gravity plane
vectorarc([grid_x(2,1)-grid_x(1,1); grid_y(2,1)-grid_y(1,1); ...
     grid_z(2,1)-grid_z(1,1)], gravnormal, gravity(:,1), 2, false, true, ...
     false, 10, [0.4 0.9 0.4]);

% Draw magnetic field vector
quiver3(B(1,1),B(2,1), B(3,1), B(1,2), B(2,2), B(3,2), 0, ...
     'color',[0.9 0.0 0.9]);


% Label the gravity and magnetic field vectors
textpos = gravity(:,1)+gravity(:, 2)+gravity(:, 2)*(1/norm(gravity(:, 2)));
H=text(textpos(1), textpos(2), textpos(3), 'G', 'color', [0.2 0.8 0.2]);
set(H, 'fontsize', 14);

textpos = B(:,1)+B(:, 2)+B(:, 2)*(1/norm(B(:, 2)));
H=text(textpos(1), textpos(2), textpos(3), 'B', 'color', [0.9 0.0 0.9]);
set(H, 'fontsize', 14);

% set white background color
set(gcf,'color','w');

hold off;



% --------------------------------------------------------------------------
% Figure 3: add angles of gravity vector to Figure 2,
%                give them the labels from AN4248
% --------------------------------------------------------------------------

figure(3);
clf;
hold on;
axis off;
axis equal;

% Set labels for Matlab/Octave coordinate X, Y & Z axes
% ( mostly for debugging purposes )
xlabel('X');
ylabel('Y');
zlabel('Z');
```

```
% Plot the rotated Earth's surface grid first so it usually gets below the
% others in handle order.
gridhandle=plot3(grid_x, grid_y, grid_z, 'color',[0.7 1.0 0.7], ...
     'linewidth',1, 'linestyle','-');

% Draw axes representing the device's coordinate system
plot3([-20;20],[0;0],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[-20;20],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[0;0],[-20;20], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');

% Label the device coordinate axes.
textpos = [18;1;1];
H=text(textpos(1), textpos(2), textpos(3), 'Y');
set(H, 'fontsize', 11);
textpos = [0.5;18;1];
H=text(textpos(1), textpos(2), textpos(3), 'X');
set(H, 'fontsize', 11);
textpos = [0.5;0.5;-18];
H=text(textpos(1), textpos(2), textpos(3), 'Z');
set(H, 'fontsize', 11);


% Draw the gravity vector. Start from origin this time.
quiver3(0,0,0, gravity(1,2), gravity(2,2), gravity(3,2), 0, ...
     'color',[0.2 0.8 0.2]);

% Right angle symbol between gravity vector and gravity plane
% Don't draw this any more, since it's just in the way of readability.
% The previous figure already illustrated it's the normal of the surface.
%vectorarc([grid_x(2,1)-grid_x(1,1); grid_y(2,1)-grid_y(1,1); ...
%    grid_z(2,1)-grid_z(1,1)], gravity(:,2), gravity(:,1), 2, false, true, ...
%    false, 10, [0.4 0.9 0.4]);

% Draw magnetic field vector.
quiver3(B(1,1),B(2,1),B(3,1), B(1,2),B(2,2),B(3,2), 0, 'color',[0.9 0.0 0.9]);

% Project gravity onto the x-z plane to illustrate the gravity rotation an-
gles.
% Normal of the x-z plane is the y axis:
xznormal = [0;-1;0];
% Project gravity onto the normal given by y-axis.
% ( v . s / s . s ) s    ( dot product is done by v' * s in Matlab )
gravity_xz = (gravity(:,2)'*xznormal) / (xznormal' * xznormal) * xznormal;

% Projection on the plane is original + negation of the projection on
% the normal:
gravity_xz = gravity(:,2) - gravity_xz;

% Draw the projection of gravity on the x-z plane.
quiver3(0,0,0, gravity_xz(1), gravity_xz(2), gravity_xz(3), 0, ...
     'color',[0.2 0.8 0.2], 'linewidth',1, 'linestyle','--');


% Phi φ is for roll around the phone's x-axis. Theta θ is for pitch around
% the phone's y-axis. Psi ψ is yaw around the z-axis, the e-compass pointing
% direction, which is our goal.
% The phone axes in AN4248 are cartesian handed,
% but x points to the phone's "top", y to the right and z downwards.
% Since I started putting the grid and gravity into negative z already,
% the axes in these illustrations are not going to match with mobile phone
% coordinate system.
% Draw angles phi φ and theta θ:
vectorarc(gravity_xz,gravity(:,2),[0;0;0],10, '\theta', true, false,10);
```

```
vectorarc([0;0;-1],gravity_xz,[0;0;0],10, '\phi', true, false,10);


% Label the gravity and magnetic field vectors
textpos = gravity(:,1)+gravity(:, 2)+gravity(:, 2)*(1/norm(gravity(:, 2)));
H=text(textpos(1), textpos(2), textpos(3), 'G', 'color', [0.2 0.8 0.2]);
set(H, 'fontsize', 14);

textpos = B(:,1)+B(:, 2)+B(:, 2)*(1/norm(B(:, 2)));
H=text(textpos(1), textpos(2), textpos(3), 'B', 'color', [0.9 0.0 0.9]);
set(H, 'fontsize', 14);

% set white background color
set(gcf,'color','w');

% zoom closer so the angles are clearly visible.
zoom(1.5);
hold off;




% --------------------------------------------------------------------------
% Figure 4: a new magnetic vector appears, with the angles from gravity having
%               moved between old and new magvectors now
% --------------------------------------------------------------------------


figure(4);
clf;
hold on;
axis off;
axis equal;

% Set labels for Matlab/Octave coordinate X Y & Z axes
% ( mostly for debugging purposes )
xlabel('X');
ylabel('Y');
zlabel('Z');


% Plot the rotated Earth's surface grid first so it usually gets below the
% others in handle order.
gridhandle=plot3(grid_x, grid_y, grid_z, ...
     'color',[0.7 1.0 0.7], 'linewidth',1, 'linestyle','-');


% Draw axes representing the device's coordinate system
plot3([-20;20],[0;0],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[-20;20],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[0;0],[-20;20], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');

% Label the device coordinate axes.
textpos = [18;1;1];
H=text(textpos(1), textpos(2), textpos(3), 'Y');
set(H, 'fontsize', 11);
textpos = [0.5;18;1];
H=text(textpos(1), textpos(2), textpos(3), 'X');
set(H, 'fontsize', 11);
textpos = [0.5;0.5;-18];
H=text(textpos(1), textpos(2), textpos(3), 'Z');
set(H, 'fontsize', 11);


% Draw the gravity vector. Start from origin this time.
```

```
quiver3(0,0,0, gravity(1,2), gravity(2,2), gravity(3,2), 0, ...
     'color',[0.2 0.8 0.2]);

% Right angle symbol between gravity vector and gravity plane
% Don't draw this any more, since it's just in the way of readability.
% The previous figure already illustrated it's the normal of the surface.
%vectorarc([grid_x(2,1)-grid_x(1,1); grid_y(2,1)-grid_y(1,1); ...
%    grid_z(2,1)-grid_z(1,1)], gravity(:,2), gravity(:,1), 2, false, true, ...
%    false, 10, [0.4 0.9 0.4]);

% Draw magnetic field vector.
quiver3(B(1,1),B(2,1), B(3,1), B(1,2), B(2,2), B(3,2), 0, ...
     'color',[0.9 0.0 0.9]);

% Project gravity onto the XZ plane to illustrate the gravity rotation angles.
% Normal of the X-Z plane is the Y axis:
xznormal = [0;-1;0];
% Project gravity onto the normal given by y-axis.
% ( v . s / s . s ) s    ( dot product is done by v' * s in Matlab )
gravity_xz = (gravity(:,2)'*xznormal) / (xznormal' * xznormal) * xznormal;

% Projection on the plane is original + negation of the projection on the
% normal:
gravity_xz = gravity(:,2) - gravity_xz;

% Draw the projection of gravity on the x-z plane.
quiver3(0,0,0, gravity_xz(1), gravity_xz(2), gravity_xz(3), 0, ...
     'color',[0.2 0.8 0.2],'linewidth',1, 'linestyle','--');

% Phi φ is for roll around the phone's x-axis. Theta θ is for pitch around
% the phone's y-axis. Psi ψ is yaw around the z-axis, the e-compass pointing
% direction, which is our goal.
% The phone axes in AN4248 are cartesian handed,
% but x points to the phone's "top", y to the right and z downwards.
% Since I started putting the grid and gravity into negative z already,
% the axes in these illustrations are not going to match with mobile phone
% coordinate system.
% Draw angles phi φ and theta θ:
vectorarc(gravity_xz,gravity(:,2),[0;0;0],10, '\theta', true, false,10);
vectorarc([0;0;-1],gravity_xz,[0;0;0],10, '\phi', true, false,10);


% Draw a matching projection and rotations for the magnetic field too:
% This time we can't just project onto an axis plane, but we need to
% apply real rotation angles to make the magnetism move like it should.

% Calculate real phi φ and theta θ values from the rotated gravity vectors.
% Vector angle calculation formula is from
% https://se.mathworks.com/matlabcentral/answers/16243-angle-between-two-vec-
tors-in-3d
phi = atan2(norm(cross([0;0;-1],gravity_xz)), dot([0;0;-1],gravity_xz));
theta = atan2(norm(cross(gravity_xz,gravity(:,2))), ...
     dot(gravity_xz,gravity(:,2)));

% Rotate B with theta around the y axis to get the intermediary vector after
% the 1st rotation.
rotatedB1 = rotateZ(-theta)*B(:,2);
Bg = rotateX(-phi)*rotatedB1;

% Draw the rotated B vectors
quiver3(0,0,0, rotatedB1(1), rotatedB1(2), rotatedB1(3), 0, ...
     'color',[0.9 0.0 0.9],'linewidth',1, 'linestyle','--');
quiver3(0,0,0, Bg(1), Bg(2), Bg(3), 0, ...
     'color',[0.4 0.0 0.9],'linewidth',1, 'linestyle','-');
```

```
% Phi φ and theta θ for the magnetism vector:
vectorarc(rotatedB1,B(:,2),[0;0;0],8, '\theta', true, false,10);
vectorarc(rotatedB1,Bg,[0;0;0],10, '\phi', true, true,10);



% Label the gravity and magnetic field vectors
textpos = gravity(:,1)+gravity(:, 2)+gravity(:, 2)*(1/norm(gravity(:, 2)));
H=text(textpos(1), textpos(2), textpos(3), 'G', 'color', [0.2 0.8 0.2]);
set(H, 'fontsize', 14);

textpos = B(:,1)+B(:, 2)+B(:, 2)*(1/norm(B(:, 2)));
H=text(textpos(1), textpos(2), textpos(3), 'B', 'color', [0.9 0.0 0.9]);
set(H, 'fontsize', 14);

textpos = B(:,1)+Bg+Bg*(1/norm(Bg));
H=text(textpos(1), textpos(2), textpos(3), 'Bg', 'color', [0.4 0.0 0.9]);
set(H, 'fontsize', 14);

% set white background color
set(gcf,'color','w');

hold off;



% --------------------------------------------------------------------------
% Figure 5: Old magvector is gone, the realigned magvector has now a
% projection on the coord plane, and angle theta shown between them.
% --------------------------------------------------------------------------


figure(5);
clf;
hold on;
axis off;
axis equal;

% Set labels for Matlab/Octave coordinate X, Y & Z axes
% ( mostly for debugging purposes )
xlabel('X');
ylabel('Y');
zlabel('Z');


% Plot the rotated Earth's surface grid first so it usually gets below the
% others in handle order.
% gridhandle=plot3(grid_x, grid_y, grid_z, 'color',[0.7 1.0 0.7], ...
%     'linewidth',1, 'linestyle','-');

% This time we plot the unrotated surface grid to demonstrate that the
% magnetism vector was rotated with the gravity and Bg is the same against the
% unrotated grid as B was against the rotated grid.
gridhandle=plot3(flat_grid_x, flat_grid_y, flat_grid_z, ...
     'color',[0.9 0.7 1.0], 'linewidth',1, 'linestyle','-');


% Draw axes representing the device's coordinate system
plot3([-20;20],[0;0],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[-20;20],[0;0], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');
plot3([0;0],[0;0],[-20;20], 'color',[0 0 0], 'linewidth',1, 'linestyle','--');

% Label the device coordinate axes.
textpos = [18;1;1];
```

```
H=text(textpos(1), textpos(2), textpos(3), 'Y');
set(H, 'fontsize', 11);
textpos = [0.5;18;1];
H=text(textpos(1), textpos(2), textpos(3), 'X');
set(H, 'fontsize', 11);
textpos = [0.5;0.5;-18];
H=text(textpos(1), textpos(2), textpos(3), 'Z');
set(H, 'fontsize', 11);



% Project the gravity corrected magnetism vector onto the x-y plane to get the
% final angle psi ψ: yaw around the z-axis, the e-compass pointing direction.

% Normal of the x-y plane is the z axis:
xynormal = [0;0;1];
% Project gravity onto the normal
Bxy = (Bg'*xynormal) / (xynormal' * xynormal) * xynormal;

% Projection on the plane is original + negation of the projection on
% the normal:
Bxy = Bg - Bxy;


% Draw the gravity-corrected magnetic field vector.
quiver3(0,0, 0, Bg(1), Bg(2), Bg(3), 0, 'color',[0.4 0.0 0.9]);

% Draw the final projection of magnetism on the x-y plane.
quiver3(0,0,0, Bxy(1), Bxy(2), Bxy(3), 0, 'color',[0.0 0.0 0.9], ...
     'linewidth',1, 'linestyle','-');


% Label the magnetic field vectors
textpos = B(:,1)+Bg+Bg*(1/norm(Bg));
H=text(textpos(1), textpos(2), textpos(3), 'Bg', 'color', [0.4 0.0 0.9]);
set(H, 'fontsize', 14);

textpos = B(:,1)+Bxy+Bxy*(1/norm(Bxy));
H=text(textpos(1), textpos(2), textpos(3), 'Bxy', 'color', [0.0 0.0 0.9]);
set(H, 'fontsize', 14);


% Psi ψ yaw arc is against the device x-axis.
% ( y-axis of our Matlab/Octave environment)
vectorarc(Bxy,[0;1;0],[0;0;0],10, '\psi', true, false,10);


% set white background color
set(gcf,'color','w');

hold off;




% ------------------------------------------------------------------------
% Figure 6: Phone coordinate system. Just a box depicting the device,
%              and coordinate axes aligned with the box.
% ------------------------------------------------------------------------


figure(6);
clf;
```

```
hold on;
axis off;
axis equal;

% Set labels for Matlab/Octave coordinate X, Y & Z axes
% ( mostly for debugging purposes )
xlabel('X');
ylabel('Y');
zlabel('Z');


% Draw axes representing the device's coordinate system
plot3([-20;20],[0;0],[0;0], 'color',[0 0 0], 'linewidth',2, 'linestyle','--');
plot3([0;0],[-20;20],[0;0], 'color',[0 0 0], 'linewidth',2, 'linestyle','--');
plot3([0;0],[0;0],[-20;20], 'color',[0 0 0], 'linewidth',2, 'linestyle','--');

% Label the device coordinate axes.
textpos = [18;1;1];
H=text(textpos(1), textpos(2), textpos(3), 'Y');
set(H, 'fontsize', 15);
textpos = [0.5;18;1];
H=text(textpos(1), textpos(2), textpos(3), 'X');
set(H, 'fontsize', 15);
textpos = [0.5;0.5;-18];
H=text(textpos(1), textpos(2), textpos(3), 'Z');
set(H, 'fontsize', 15);


% Draw the mobile device as a black box with a screen
hl = 8; % 1/2 dimensions of the box
hw = 4;
hh = 1;
scr_m = 1;
scr_h = 12;

phone1=[-hw -hw +hw +hw -hw;
        -hl +hl +hl -hl -hl;
        +hh +hh +hh +hh +hh];
phone2=[-hw -hw +hw +hw +hw +hw +hw;
         -hl -hl -hl -hl -hl +hl +hl;
         +hh -hh -hh +hh -hh -hh +hh];
phone3=[-hw -hw -hw -hw +hw;
         -hl +hl +hl +hl +hl;
         -hh -hh +hh -hh -hh];

phone4=[-hw+scr_m +hw-scr_m +hw-scr_m       -hw+scr_m       -hw+scr_m;
         +hl-scr_m +hl-scr_m +hl-scr_m-scr_h +hl-scr_m-scr_h +hl-scr_m;
         +hh       +hh       +hh              +hh              +hh];

plot3(phone1(1,:),phone1(2,:),phone1(3,:), 'color',[0.3 0.3 0.3], ...
     'linewidth',1, 'linestyle','-');
plot3(phone2(1,:),phone2(2,:),phone2(3,:), 'color',[0.3 0.3 0.3], ...
     'linewidth',1, 'linestyle','-');
plot3(phone3(1,:),phone3(2,:),phone3(3,:), 'color',[0.8 0.8 0.8], ...
     'linewidth',1, 'linestyle','--');
plot3(phone4(1,:),phone4(2,:),phone4(3,:), 'color',[0.3 0.3 0.3], ...
     'linewidth',1, 'linestyle','-');

% set white background color
set(gcf,'color','w');

zoom(1.5);
hold off;
```

```
% ------------------------------------------------------------------------
% Figure 7: Calibration as vector sum. This is a 2D illustration, since that
%               is enough to demonstrate the point of measurements being a sum
%               of hard iron and Earth magnetism.
% ------------------------------------------------------------------------


figure(7);
clf;
hold on;
axis off;
axis equal;

% Set labels for Matlab/Octave coordinate X, Y & Z axes
% ( mostly for debugging purposes )
xlabel('X');
ylabel('Y');
zlabel('Z');


% Semi-randomly picked values for Earth magnetism and hard iron vectors.
% The hard iron vector is longer as in real life.
earth=[-1;-3];
hardiron=[9;2];

% Plot hard iron and Earth magnetism vectors from origin
quiver(0,0, hardiron(1), hardiron(2), 0, 'color',[0.9 0.0 0.2], ...
     'linewidth',3, 'linestyle','-');
quiver(0,0, earth(1), earth(2), 0, 'color',[0.0 0.0 0.9], ...
     'linewidth',3, 'linestyle','--');

% Earth magnetism as an addition to the hard iron vector
quiver(hardiron(1), hardiron(2), earth(1), earth(2), 0, ...
     'color',[0.0 0.0 0.9],'linewidth',3, 'linestyle','-');


% Now do a circle around the hard iron vector end,
% depicting the area of uncalibrated measurements.
theta = linspace(0, 2*pi, 32);
circle = []; % "reset" to avoid errors across multiple runs
circle(1,:) = norm(earth)*cos(theta);
circle(2,:) = norm(earth)*sin(theta);
plot(hardiron(1)+circle(1,:),hardiron(2)+circle(2,:), ...
     'color',[1.0 0.6 1.0], 'linewidth',4, 'linestyle',':');

% Another circle around the origin, to demonstrate calibration result area
% at the origin.
plot(circle(1,:), circle(2,:), 'color',[0.6 0.6 1.0], ...
     'linewidth',4, 'linestyle','-.');

% Label for the origin point
H=text(-0.3, 0.2, 'O');
set(H, 'fontsize', 24);

% set white background color
set(gcf,'color','w');

hold off;
```

Liite 2. Havainnekuvien käyttämät funktiot

Liitteen 1 piirtokoodin tarvitsemat Matlab/Octave-funktiot. Matlabin käytännön mukaisesti jokaisen funktion tulee olla omassa tiedostossaan, ja tämän tiedoston nimen tulee vastata itse funktion nimeä, päätteenään `.m`. Liitteen ohjelmakoodi sisältää neljä funktiota.

```
function [ R ] = rotateX( angle )
% rotateX(angle)
% Returns 3D non-homogenised rotation matrix for rotation around X axis.

R = [ 1 0 0; 0 cos(angle) sin(angle); 0 -sin(angle) cos(angle) ];

end


function [ R ] = rotateY( angle )
% rotateY(angle)
% Returns 3D non-homogenised rotation matrix for rotation around Y axis.

R = [ cos(angle) 0 -sin(angle); 0 1 0; sin(angle) 0 cos(angle) ];

end


function [ R ] = rotateZ( angle )
% rotateZ(angle)
% Returns 3D non-homogenised rotation matrix for rotation around Z axis.

R = [ cos(angle) sin(angle) 0; -sin(angle) cos(angle) 0; 0 0 1 ];

end



function H = vectorarc(v1, v2, centerpoint, radius, label, ...
     right, reflex, resolution, ...
     color, linewidth, linestyle, fontsize)
%
% H = vectorarc(v1, v2, centerpoint=[0;0;0], radius=1, label=false,
%    right=false, reflex=false, resolution=10,
%   color=[0.0 0.0 0.0], linewidth=1, linestyle='-', fontsize=10)
%
% Draws an arc with the given radius between input vectors 'v1' and 'v2'.
%
% The vectors give the start and end points for the arc.
% They also give the orientation of the angle arc.
% The vectors must be given as a 3 by 1 matrix and they cannot
% be multiples of each other ( same or opposite direction ).
% The 'centerpoint' must also be a 3 by 1 matrix.
% If the 'label' is a string, it is drawn next to the arc.
% If the 'label' is a boolean value, no label is drawn by vectorarc().
% If the 'label' is a scalar numeric value <= pi,
%  a label is drawn stating the angle between the vectors in radians.
% If the label is a scalar numeric value > pi,
%  a label is drawn stating the angle between the vectors in degrees.
% The number itself is not used. You must convert your number to a string
% to get it used by vectorarc(). E.g. vectorarc(...,sprintf('%f',3.14159)).
% If the label is something else, it is ignored and no label is printed.
```

```
% If 'right'==true, a right angle square is drawn instead of an arc,
%  if the vectors are right angled. The square uses 'radius' for its size.
% If 'reflex'==false, the acute, obtuse or right angle between the vectors
%  is usually drawn. If true, the reflex angle (>180 degrees) between them is
%  drawn.
%  Exception: In some quadrangles reflex==false results in a > 180 degree
%  angle! This is a bug which remains uncorrected due to time constraints.
%  The shorter angle can in those cases be obtained with reflex==true.
% 'resolution' is the number of points used for drawing the arc.
% 'color', 'linewidth', and 'linestyle' values are passed to plot3()
% as linespec properties with the same names. See documentation of
% Matlab/Octave/Freemat for description of the line drawing properties.
% 'fontsize' is a standard property for text. See corresponding documentation.
%
% Returns an array containing the handle returned by plot3(), which is called
%  internally to draw the arc, as the first member,
%  and the handle of the label text() as the second member.
%  If labels are disabled, the second return value member is 0;
% In case of error, the return value is a single boolean false value.
%  Testing for error can be done by the comparison isbool(retval).
%  Comparison retval == false is not reliable if graphics handles can be 0.
%
% Example of usage:
%
% v1 = [1;2;3];
% v2 = [4;5;6];
% figure 1;
% hold on;
%
% % ( Here you plot lines or vectors v1 and v2 for between which you
% % want to draw an angle arc for mathematical demonstration.
% % Let's assume they form an angle at point [1;1;1], and aren't aligned
% % with any coordinate axes so you can't use easy 2D math.
% % Let's also assume you want to make your demonstration compatible
% % between Matlab and Octave, so you can't use Matlab's circle() function. )
%
% vectorarc(v1, v2, [1;1;1], 1, 'alpha = 30 degrees');
% hold off;
%
% Dependencies:
% none
%
% Author:
% Jussi Nikula (2018)

% Fill in default values for unspecified optional arguments
if ( nargin < 3 )
     centerpoint=[0;0;0];
end
if ( nargin < 4 )
     radius=1;
end
if ( nargin < 5 )
     label=false;
end
if ( nargin < 6 )
     right=false;
end
if ( nargin < 7 )
     reflex=false;
end
if ( nargin < 8 )
     resolution=20;
end
if ( nargin < 9 )
```

```
     color=[0.0 0.0 0.0];
end
if ( nargin < 10 )
     linewidth=1;
end
if ( nargin < 11 )
     linestyle='-';
end
if ( nargin < 12 )
     fontsize=10;
end


% Validate input arguments:
error = false;
if ( size(v1,1) < 3 )
     error = true;
     fprintf('Error: vectorarc() input vector ''v1'' must have 3 rows!\n');
end
if ( size(v2,1) < 3 )
     error = true;
     fprintf('Error: vectorarc() input vector ''v2'' must have 3 rows!\n');
end
if ( size(centerpoint,1) < 3 )
     error = true;
     fprintf('Error: vectorarc() input ''centerpoint'' must have 3 rows!\n');
end
if ( ~isnumeric(radius) )
     error = true;
     fprintf('Error: vectorarc() input ''radius'' must be numeric!\n');
end
if ( ~islogical(right) )
     error = true;
     fprintf('Error: vectorarc() input ''right'' must be a boolean!\n');
end
if ( ~islogical(reflex) )
     error = true;
     fprintf('Error: vectorarc() input ''reflex'' must be a boolean!\n');
end
if ( ~isnumeric(resolution) )
     error = true;
     fprintf('Error: vectorarc() input ''resolution'' must be numeric!\n');
end
if ( size(color,2) < 3 )
     error = true;
     fprintf('Error: vectorarc() input ''color'' must have 3 columns!\n');
end
if ( ~isnumeric(linewidth) )
     error = true;
     fprintf('Error: vectorarc() input ''linewidth'' must be numeric!\n');
end
if ( ~ischar(linestyle) )
     error = true;
     fprintf('Error: vectorarc() input ''linestyle'' must be a string!\n');
end
if ( ~isnumeric(fontsize) )
     error = true;
     fprintf('Error: vectorarc() input ''fontsize'' must be numeric!\n');
end
if (error)
     H = false;
     return;
end

% First we need a transformation to a plane that goes along v1 and v2,
```

```
% so we can use 2D math to draw an arc.
% Using the approach described by Nosredna in
% https://stackoverflow.com/questions/1023948/rotate-normal-vector-onto-axis-
plane

% Set new coordinate system axes starting from the normal of the plane
% set up by v1 and v2. This will ensure this new system is aligned with
% v1 and v2.

new_z = cross(v2, v1); % This order puts them on the x-y plane in the same
                       % direction with the originals. Otherwise it's a mirror
                       % image, though the arc gets transformed back ok anyways.

if (norm(new_z) > -1e-12 && norm(new_z) < 1e-12)
   % Floating point fuzz safe comparison (Octave suffers from fuzz issues).
   fprintf('Error: v1 and v2 are multiples of each other\n');
   fprintf('( the same direction or opposite )!\n');
   fprintf('There can be an infinite number of planes between them\n');
   fprintf('so they are not enough to tell us an orientation for our arc.\n');
   fprintf('To draw a full circle or a semicircle, you need to call');
   fprintf(' vectorarc() twice\n');
   fprintf('and construct the desired result from two pieces.\n');
   H = false;
   return;
end
new_y = cross([1;0;0], new_z);
if ( norm(new_z) > -1e-12 && norm(new_z) < 1e-12 )
     % If the new z-axis is aligned with the current coordinate system x-axis,
     % the cross product is [0;0;0] so use the y-axis instead.
     % Floating point fuzz safe equality comparison.
     new_y = cross([0;1;0], new_z);
end
% Since new_y is always a non-zero cross product of a non-zero new_z,
% this one can never fail:
new_x = cross(new_z, new_y);

% And since new_z is checked to always be a non-zero cross product of
% v2 and v1, we always get a transformation to the v1-v2 plane where
% z component transforms to 0.
% If it's a mirror world of some sort, it doesn't matter since the
% transformation back always returns the original orientation.

% Normalise to unit vectors
new_x = new_x*(1/norm(new_x));
new_y = new_y*(1/norm(new_y));
new_z = new_z*(1/norm(new_z));

% Construct transformation matrix back to original space.
% This is easy now that we have the new system's unit vectors.
TB = [ new_x(1) new_y(1) new_z(1); new_x(2) new_y(2) new_z(2); ...
     new_x(3) new_y(3) new_z(3)];

% Transformation from original to the new would be
% TF = [ new_x(1) new_x(2) new_x(3); new_y(1) new_y(2) new_y(3); ...
%    new_z(1) new_z(2) new_z(3)]
% which is the same as inverse(TB).

% This magical vector math operation transforms v1 and v2 into the new
% coordinate system.
% z becomes 0, so we're on the x-y plane and can draw the arc with 2D math.
% It's just amazing what you can do with the various matrix products! :)
% This is from Nosredna from the abovementioned stackexchange answer.
% rotated_v1=TF*v1 would also work.
rotated_v1 = [dot(v1,new_x);dot(v1,new_y);dot(v1,new_z)];
rotated_v2 = [dot(v2,new_x);dot(v2,new_y);dot(v2,new_z)];
```

```
% Check that the lengths match ( they do! :) )
%norm(v1)
%norm(rotated_v1)

% Check that the angle between v1 and v2 matches ( it does! :) )
% This is from
% https://se.mathworks.com/matlabcentral/answers/16243-angle-between-two-vec-
tors-in-3d
%arcrange1 = atan2(norm(cross(v1,v2)), dot(v1,v2))
%arcrange2 = atan2(norm(cross(rotated_v1,rotated_v2)), ...
%     dot(rotated_v1,rotated_v2))


% Now that we have pseudo-2D vectors with the correct angles, we can start
% building an arc with 2D math using the formula from
% https://stackoverflow.com/questions/25594597/drawing-an-arc-in-oc-
tave#25594757

% 2D bearing angles of rotated v1 and v2:
theta1 = atan2(rotated_v1(2),rotated_v1(1));
theta2 = atan2(rotated_v2(2),rotated_v2(1));

no_theta = false;

if ( dot(rotated_v1,rotated_v2) > -1e-12 ...
     && dot(rotated_v1,rotated_v2) < 1e-12 ...
     && right == true && reflex == false)
     % The angle is a right angle ( dot product == 0 ):
     % Optionally draw a right angle square between the vectors
     % instead of the arc for acute and obtuse angles.
     % Dot product is compared in a floating point safe manner
     % since Octave suffers from floating point fuzz issues with these.

     % Since the vectors are right angled, we can easily get
     % the bearings from each other.

     % Unit vectors of rotated_v1 and rotated_v2
     v1u = rotated_v1*(1/norm(rotated_v1));
     v2u = rotated_v2*(1/norm(rotated_v2));

     % The final size is obtained by multiplying the unit vectors with
     % the desired radius.
     square1 = radius*v1u;
     square2 = radius*v2u;

     % Use the same variable for storing points so we can use common
     % transformation and plotting code afterwards.
     xyarc = [square1 square1+square2 square2];

     no_theta = true;
else

     % Using extra input arg for reflex angles:
     % ( Yet another input arg is not nice, but usually the intent is probably
     %    to draw the shortest path between the vectors and their order may be
     %    difficult to know beforehand, so I'm not sure which way is better.
     %    Probably better to require stating the intent specifically. )
     if ( reflex == true )
          % Instead of the "shortest" angle between v1 and v2,
          % draw the reflex angle if told to do so.
          % linspace() always takes the shortest path between the two args so
          % we need to cheat it a little to draw a reflex angle.

          % Complement angle of theta1..theta2 difference:
```

```
            reflextheta = 2*pi-abs(theta2-theta1);

            if ( theta1 < theta2 )
                theta = linspace(theta1-reflextheta, theta1, resolution);
            else
                theta = linspace(theta2-reflextheta, theta2, resolution);
            end
        else
            % Generate 'resolution' amount of intermediary angles
            % that are between theta1 and theta2:
            theta = linspace(theta1, theta2, resolution);
        end

        % Reflex angle depends on input vector order:
        % ( It's nice to have one less input arg, but usually the intent is
        %    probably to draw the shortest path between the vectors and their
        %    order may be difficult to know beforehand, so I'm not sure which
        %    way is better... )
        %if ( theta1 < 0 || theta2 < 0 )
        %    % One vector order results in negative angles and the other in
        %    % positive angles. theta2-theta1 is always the same sign and theta1
        %    % is always less than theta2.
        %    % This is due to the vector transformation working almost too well.
        %    % This needs more testing: In some quadrangles theta2 < theta1!
        %    % linspace() always takes the shortest path between the two args so
        %    % we need to cheat it a little to draw a reflex angle.
        %
        %    % Complement angle of theta1..theta2 difference:
        %    reflextheta = 2*pi-abs(theta2-theta1);
        %
        %    theta = linspace(theta1-reflextheta, theta1, resolution);
        %else
        %    % Generate 'resolution' amount of intermediary angles
        %    % that are between theta1 and theta2:
        %    theta = linspace(theta1, theta2, resolution);
        %end

        % Generate coords for an arc between angles theta1 and theta2 and center
        % at origin.
        xyarc(1,:) = radius*cos(theta);
        xyarc(2,:) = radius*sin(theta);
        xyarc(3,:) = linspace(0,0,size(xyarc(1,:),1)); % Add z-coord 0

end

% Transform the arc to the original alignment of v1 and v2:
arc = TB*xyarc;

% We still need to add the centerpoint to each arc coord
% to draw it where it belongs:
arc = centerpoint + arc;

% Calculate position for the label text, using the centermost arc point
% as a reference:
if ( no_theta )
    % Calculate text position from theta1 and theta2 since we have no
    % 'theta' defined in this case.
    texttheta = theta1+(theta2-theta1)/2;
else
    % Use theta for text position, so we always get the same place with the
    % arc.
    texttheta = theta(round(size(theta,2)/2));
end
textpos(1,1) = (radius+0.5+0.1*radius)*cos(texttheta);
textpos(2,1) = (radius+0.5+0.1*radius)*sin(texttheta);
```

```
textpos(3,1) = 0;

% Transform the text position to 3D space just like the arc points:
textpos = TB*textpos;
textpos = centerpoint + textpos;


% Graphical debugging code:
%figure 1;
%clf;
%hold on;
%xlabel('X')
%ylabel('Y')
%zlabel('Z')
%axis equal;
%grid on;
%plot3(xyarc(1,:), xyarc(2,:), xyarc(3,:), ...
%     'color',[0.4 1.0 0.4], 'linewidth',1);


% Draw the arc:
H(1)=plot3(arc(1,:), arc(2,:), arc(3,:), ...
     'color', color, 'linewidth', linewidth, 'linestyle', linestyle);


% Draw the label if asked to do so:
if islogical(label)
     % If 'label' is a boolean, don't print a label. We just check for this
     % type first in the if-else chain since booleans are also scalars.
     H(2) = 0;
elseif ischar(label)
     % Print user supplied label text.
     H(2)=text(textpos(1), textpos(2), textpos(3), label, 'color', color, ...
          'linewidth', linewidth, 'linestyle', linestyle, 'fontsize', font-
size);
elseif isscalar(label) && isnumeric(label)
     % Degree signs and such are not very compatible, so we dare not use them
     % here. On my Windows 10, the degree sign copied from Character Map into
     % Notepad++ and this code produced a familiar extraneous unicode byte
     % artifact to the label.
     % Windows uses UTF16 and Linux uses UTF8 by default, and Matlab and
     % Octave use different output systems for plotting.
     % The environment used for testing this was Octave 4.2.2.
     if ( label <= pi )
          H(2)=text(textpos(1), textpos(2), textpos(3), ...
               sprintf('%d',abs(theta2-theta1)), ...
               'color', color, 'linewidth', linewidth, ...
               'linestyle', linestyle, 'fontsize', fontsize);
     else
          H(2)=text(textpos(1), textpos(2), textpos(3), ...
               sprintf('%.1f',abs(theta2-theta1)/pi*180), ...
               'color', color, 'linewidth', linewidth, ...
               'linestyle', linestyle, 'fontsize', fontsize);
     end
end

% More graphical debugging stuff:

%quiver3(0,0,0,v1(1),v1(2), v1(3), 0, 'color',[0.0 0.0 0.9]);
%quiver3(0,0,0,v2(1),v2(2), v2(3), 0, 'color',[0.9 0.0 0.0]);
%quiver3(0,0,0,new_x(1), new_x(2), new_x(3), 0, 'color',[0.9 0.9 0.5]);
%quiver3(0,0,0,new_y(1), new_y(2), new_y(3), 0, 'color',[0.9 0.9 0.5]);
%quiver3(0,0,0,new_z(1), new_z(2), new_z(3), 0, 'color',[0.9 0.9 0.1]);
%quiver3(0,0,0,rotated_v1(1), rotated_v1(2), rotated_v1(3), 0, ...
%     'color',[0.0 0.9 0.0]);
```

```
%quiver3(0,0,0,rotated_v2(1), rotated_v2(2), rotated_v2(3), 0, ...
%    'color',[0.0 0.9 0.0]);
%quiver3(v1(1),v1(2),v1(3),back_v1(1), back_v1(2), back_v1(3), 0, ...
%    'color',[0.9 0.0 0.0]);

% Check for the transformation back to normal space
%back_ux = TB * new_x;
%back_uy = TB * new_y;
%back_uz = TB * new_z;
%quiver3(0,0,0,back_ux(1), back_ux(2), back_ux(3), 0, 'color',[0.9 0.0 0.9]);
%quiver3(0,0,0,back_uy(1), back_uy(2), back_uy(3), 0, 'color',[0.9 0.0 0.9]);
%quiver3(0,0,0,back_uz(1), back_uz(2), back_uz(3), 0, 'color',[0.9 0.0 0.9]);

%hold off;

end
```