

Robottiohjelmoinnin harjoitustyö

Ristinollarobotti

Aaro Salosensaari

aaro.salosensaari@cs.helsinki.fi

Tietojenkäsittelytieteen laitos
Helsingin Yliopisto

Helsinki, 11. tammikuuta 2015

Sisältö

1 Ristinollarobotin kuvaus	2
2 Robotin rakenne ja toiminta	2
2.1 Materiaalit ja tarvikkeet	2
2.2 Piirtobotti	3
2.2.1 Strategiset mitat	3
2.2.2 Rakenne	3
2.2.3 Ruksin piirtäminen	4
2.3 Kamera ja kuvantunnistus	8
2.3.1 Kameran sijoittaminen	8
2.3.2 Kuvantunnistuksen toimintaidea	8
3 Ohjelmakoodi	13
3.1 PenBot	13
3.1.1 Huomioita	13
3.2 BotGame	13
3.2.1 Huomioita	15
4 Testaus	15
4.1 Ohjelmallisia testi'skriptejä'	15
4.1.1 PenBotin kynänliikuttelun testaus ja säätö	15
4.1.2 Kuvanhahmotuksen testaus ja säätö	16
4.1.3 Käskyjen välittäminen tietokoneelta PenBotille ja piirtäminen	16
4.2 Testibotit	16
4.2.1 Hello Ironman!	16
4.2.2 Hello BT!	17
4.3 Muita testiskenaarioita	17
4.3.1 Hätipysäytys	17
5 Rajoitukset ja tulevaisuus	17
5.1 Toteuttamatta jääneet ominaisuudet	17
5.2 Muita puutteita ja rajoitteita:	18
6 Käyttöohjeet	18
6.1 Ohjelmistojen käänäminen ja lataaminen NXT:lle	18
6.1.1 PenBot	18
6.1.2 BotGame	18
6.2 Pelaaminen	19

Tämä on joulun 2014 robottikurssin ([GitHub](#)) loppuraportti. Luettavin versio on [pdf](#).

1 Ristinollarobotin kuvaus



Kuva 1: Ristinollarobotti

Ristinollarobotti on ristinollaan web-kameran avulla pelaava Lego Mindstorms -robotti.

Robotti koostuu varsinaisesta piirtorobotista (PenBot) ja erillisestä tietokoneella (= kannettava tietokone) ajettavasta varsinaisesta peliohjelmasta (BotGame). Ohjelmointikieli on molemmissa Java (piirtobotissa [LeJOS](#)), ja kuvantunnistukseen käytetään [OpenCV](#):n Java-API:ia.

Peliohjelma osaa tunnistaa ulkoisen web-kameran avulla paperille piirretyt pellilaudan ja pelaajien (robotti ja sen vastustaja) sille piirtämät merkit, ja tämän avulla pelata ristinollaan ihmisvastustajaa vastaan. Peliohjelma lähettää tekoälyn valitsemat siirrot Bluetoothin yli piirtorobottiille, jolla on valmiit rutuinit ruksin piirtämisksi kuhunkin ruutuun.

2 Robotin rakenne ja toiminta

2.1 Materiaalit ja tarvikkeet

Robotin toteuttamiseen käytettiin

- (hieman vajaa) Lego Mindstorms NXT -sarja

- NXT Brick, kolme moottoria
- sekalaisia Mindstorms -legoja
- (NXT:n omia sensoreita ei tarvittu / käytetty)
- Web-kamera: Logitech C270
- Tussikynä: Stabilo Pen 68
- Kannettava tietokone jossa Bluetooth
- Kuminauhaa, paperia, teippiä, korotettu alusta kameralle

2.2 Piirtobotti

2.2.1 Strategiset mitat

- Renkaiden välinen etäisyys ~120 mm
- Etäisyys keulasta perään ~210 mm
- Säkkäkorkeus ~75 mm
- Kynän kärjen etäisyys akselista piirtoasennossa ~70 mm

2.2.2 Rakenne

Piirtobotin perusrunko perustuu [NXTPrograms.com 3-Motor Chassis](#)-rakenteeseen, jota jouduttiin hieman muokkaamaan osien puutteesta johtuen (esim. samanlaista rullapyörää ei ollut käytettävissä, joten piti soveltaa) ja johon lisättiin ylös ja alas liikkiva kynä.

Rakenteeltaan robotti on kahden moottorin avulla liikkuva auto, joka pystyy kääntymään säätelemällä vasemman ja oikeaan pyörän moottoreiden pyörimisnopeutta. Pyörittämällä moottoreita samalla nopeudella eri suuntiin robotti pystyy kääntymään paikoillaan renkaiden välisen kuvaannollisen ‘akselin’ keskipisteen ympäri.

Koska ns. rullapyöräksi sopivia pieniä renkaita ei ollut mukana allekirjoittaneelle päätyneessä sarjassa, ja käytettävissä olevien suurehkkojen kumipyörien sijoittaminen perusrunkoon osoittautui varsin haastavaksi (kumirenkaan kanssa pyörät olivat aivan liian suuria ja ilman kumiosaa sylinterimäisten pyörien liike oli liian tökkivää piirtämiseen), robotissa ei ole tukena tällaisessa Lego-autossa tavanomaisista rullapyörää (*castor wheel*), vaan yksinkertainen pyödän pintaa pitkin liukuva tuki.

Kynän liikuttelumekanismi käyttää kolmatta moottoria kynän nostamiseen ja laskemiseen paperille piirtämistä varten. Käytännössä kynä on kiinnitetty kumilenkillä telineeseen, jonka liike on rajoitettu ylös-alas -suuntaiseksi kiskojen avulla; telineeseen on kiinnitetty tapit, joiden avulla moottoriin kiinnitetyt varret voivat painaa kynän alas tai kohottaa sen ylös. Lisäksi kynän edessä ja takana on kynän

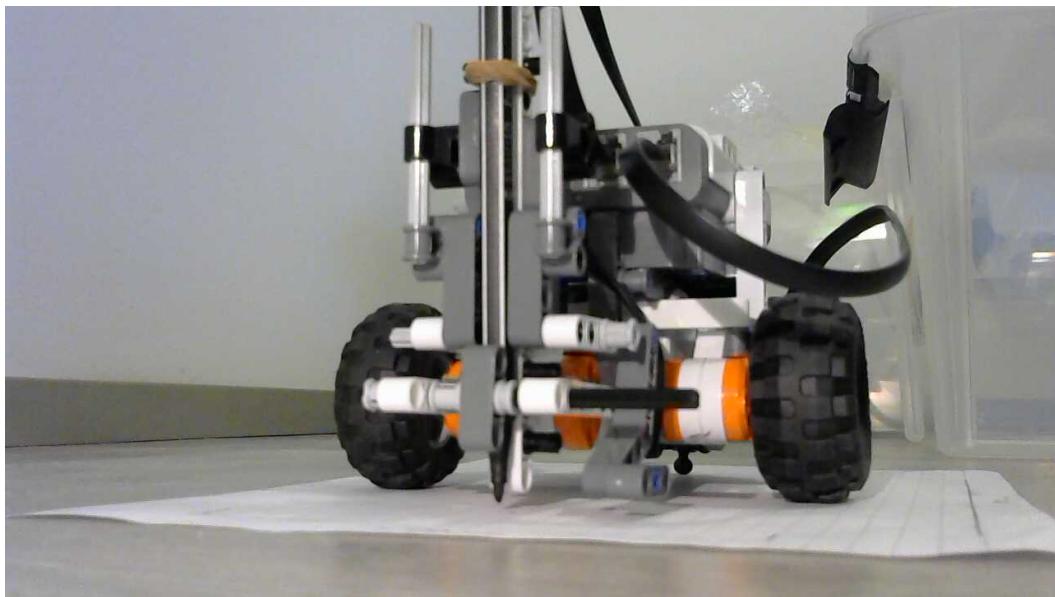
heilumista rajoittavat tuet, jotka pitävät kynän tukevasti paikoillaan piirtämisen aikana.

[Video kynän liikuttelumekanismin toiminnasta.](#)

Kumilenkkikiinnitys mahdollistaa teoriassa pienen häitävaran siltä varalta että käyttäjä menee ja poistaa turvarajat PenBotin ohjelmakoodista ja kalibroi moottorin väärin: jos moottori yrittäisi painaa kynää alemmas kuin turvalliseen käyttöön on suunniteltu, kumilenkit teoriassa joustaisivat sen sijaan että moottoriin tai rakenteeseen kohdistuisi haitallista rasitusta. (Robotin toiminta voidaan myös välittömästi keskeyttää häitäseis-nappulaa painamalla.)

Lisäksi kynän vierässä on pieni työkalu, joka helpottaa robotin asettamista oikeaan suuntaan ruutupaperin päälle.

Varsinaisen erillisen rakennusohjeen sijasta lukijaa pyydetään seuraamaan NXT-Programs.com:n [ohjeen](#) vaiheita 1 – 4 ja 12 –, ja vertailemaan eroavaisuuksien kohdalla alla oleviin kuviin. Huom. erityisesti että rullapyörän korvaavan tuen kiinnitys on erilainen.



Kuva 2: Edestä.

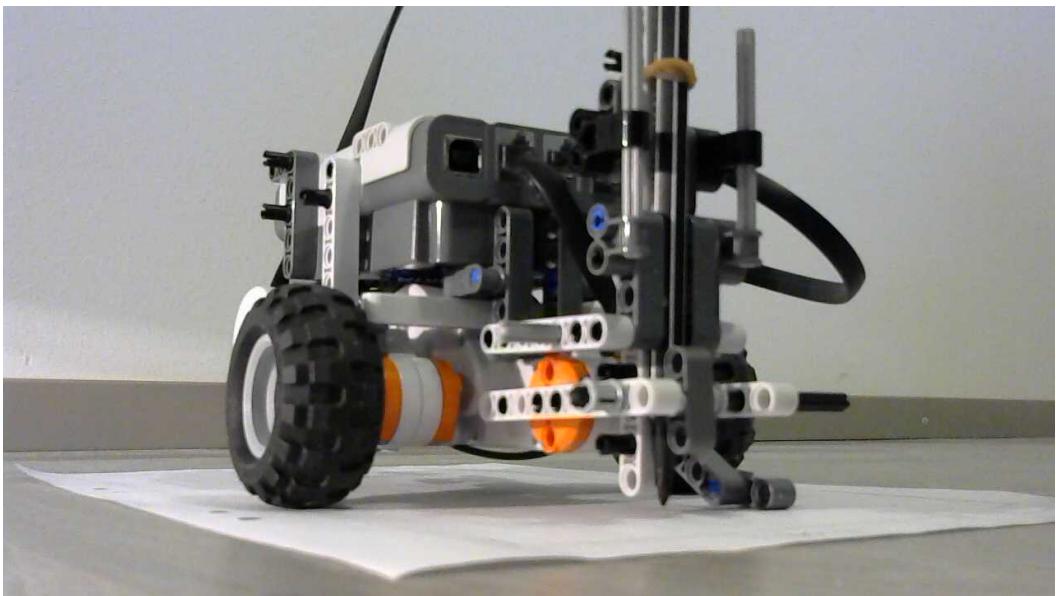
2.2.3 Ruksin piirtäminen

Robotti osaa piirtää pelilaudalle ruksin edestakaisella liikkeellä ja sivusuuntaisilla käänöksillä.

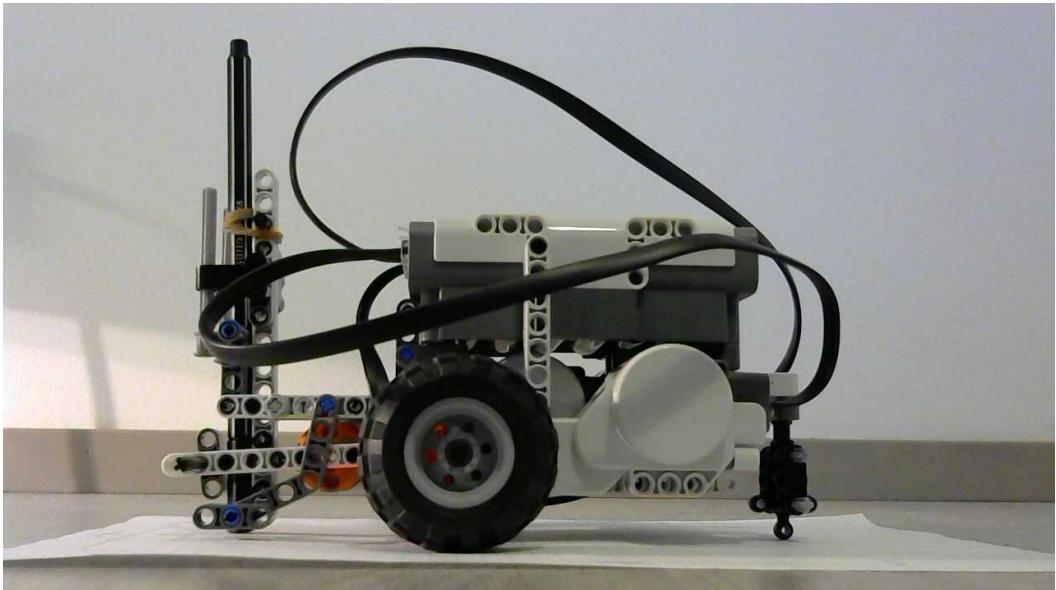
[Video: Robotti piirtää yhden ruksin.](#)

[Video: Robotti piirtää toisen ruksin.](#)

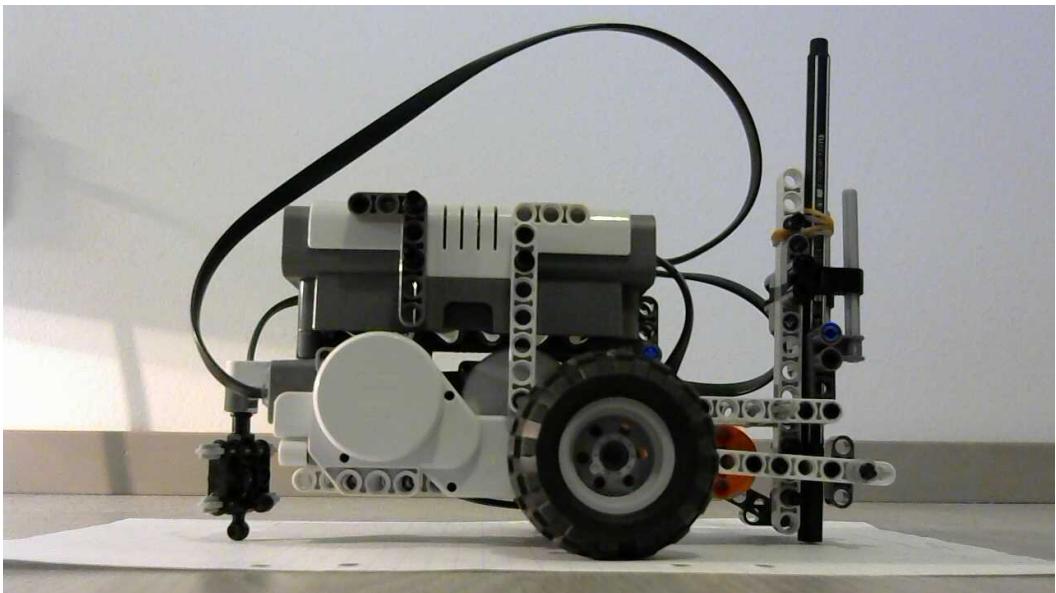
[Video: Väärin suunnattu robotti piirtää ruksin hieman sivuun ruudun keskipisteestä.](#)



Kuva 3: Toinen kuva edestä (kynämekanismi)



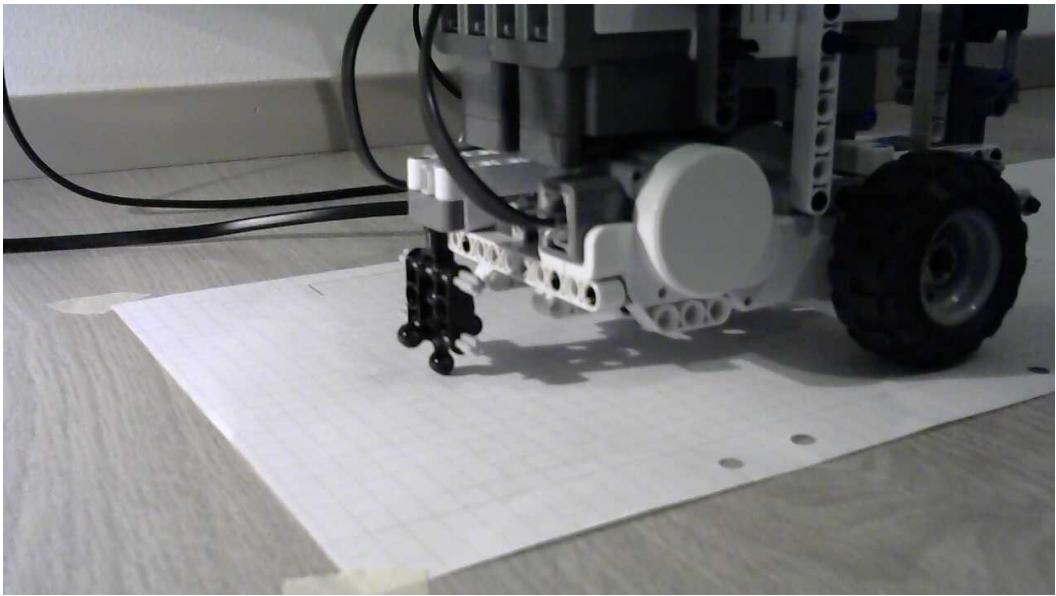
Kuva 4: Sivusta.



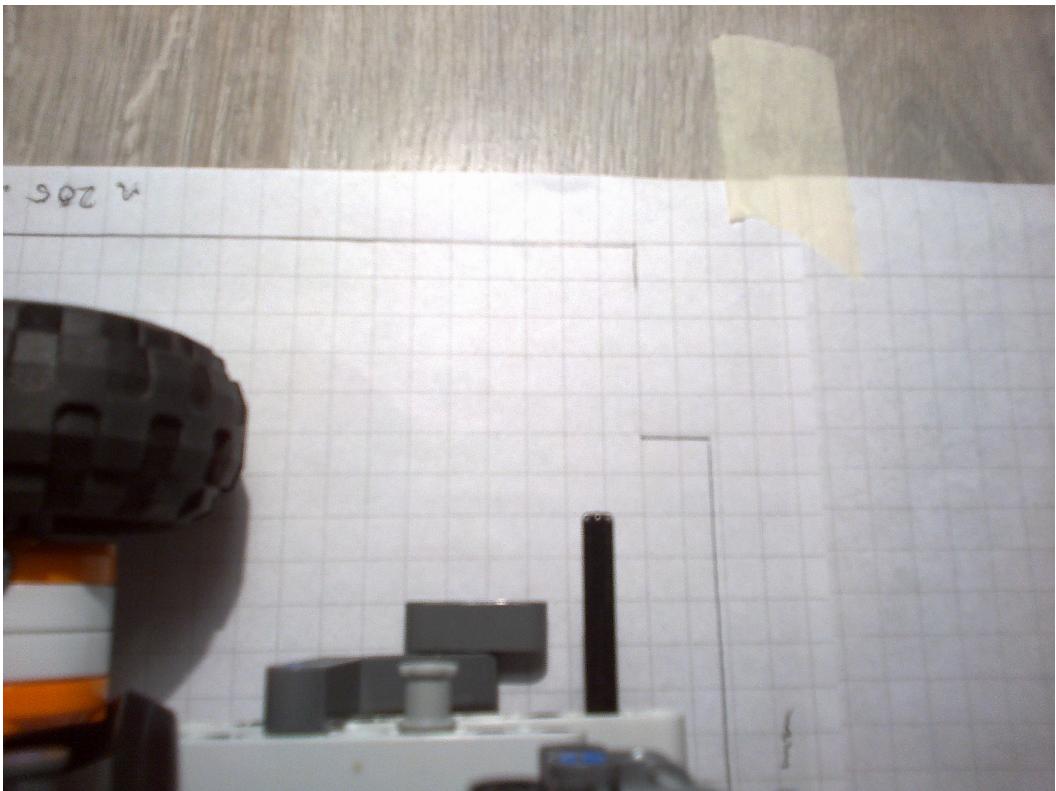
Kuva 5: Toiselta sivulta.



Kuva 6: Takaan.



Kuva 7: Peräosa

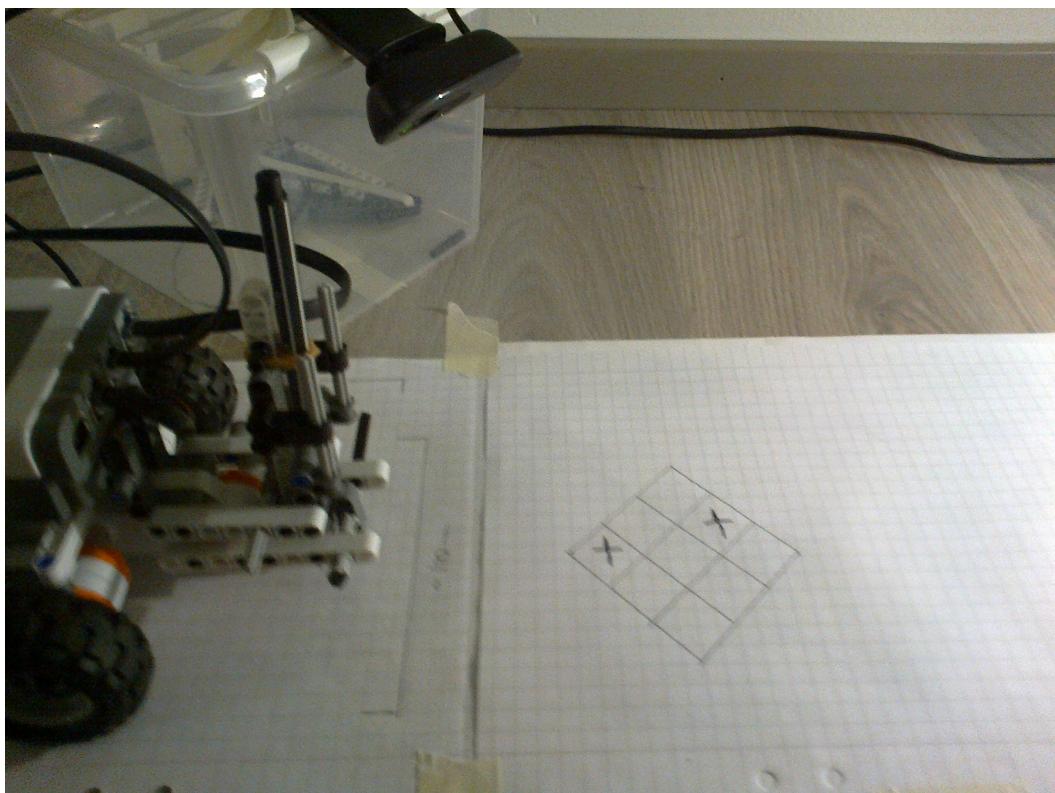


Kuva 8: Robotin asennon tarkistaminen.

2.3 Kamera ja kuvantunnistus

2.3.1 Kameran sijoittaminen

Ristinollapelikentän näkevä kamera on sijoitettu (perästä päin katsoen) robotin vasemmalle puolelle noin 15 cm korkealle alustalle. BotGame:n kuvankäsittelyrutiinit osaavat tehdä kameran kuvalle perspektiivikorjaukseen, mutta tunnistuksen luotetavuuden kannalta on suotavaa että kuva peliruudukosta ei ole liiaksi väärystynyt. Erityisesti vaaditaan että kameran näköpiirissä on mahdollisimman vähän muita mahdollisesti pysty- tai vaakasuuntaisilta viivoilta näyttäviä kohteita (paperin reuna, varjot, jne) jotka saattavat erehdyttää BotGame:n pelialueen hahmotusmetodeja.

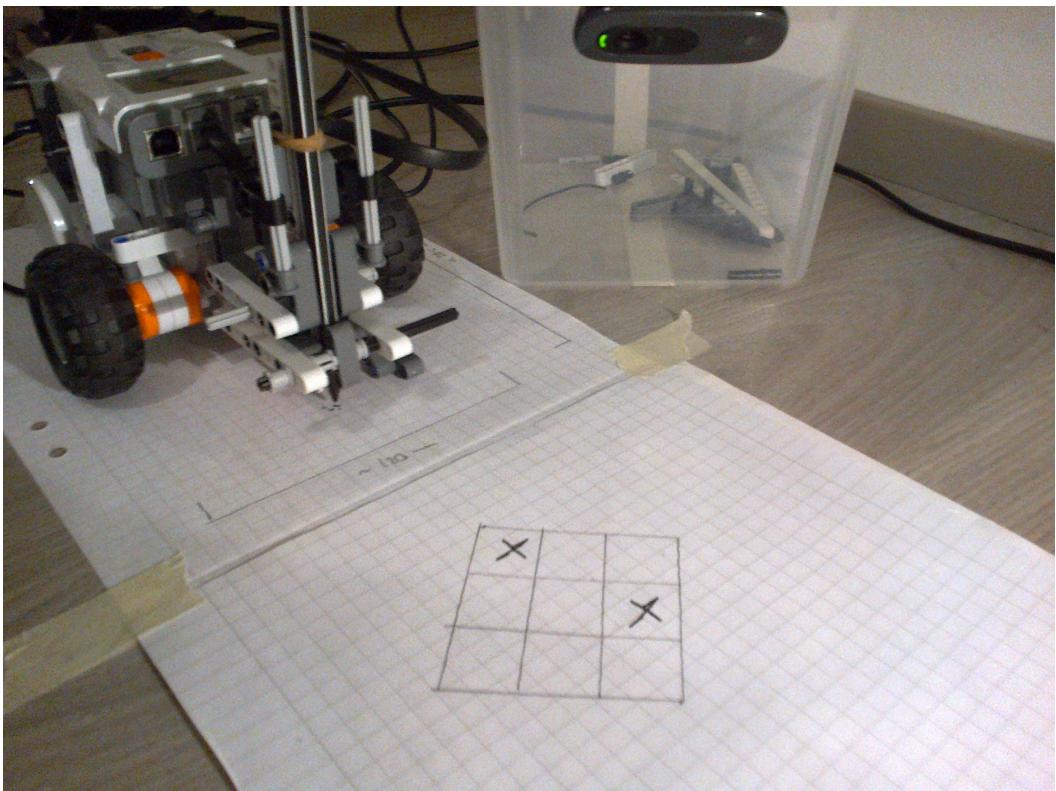


Kuva 9: Kameran asettelu, sivukuva

2.3.2 Kuvantunnistuksen toimintaidea

Kuvantunnistusmenetelmän päätähtävä oli [AI Shackin Sudoku-lukija](#), jota tosin on sovellettu varsinkin paljon. Menetelmän idea on yleisellä tasolla seuraava:

1. Ensin etsitään taustakuva, johon mahdollisia muutoksia verrataan:
2. Muunnetaan kuva harmaasävykuvaksi.
3. Ruutupaperin ruutujen häivyttämiseksi sumennetaan kuva Gauss-sumennoksella, jonka jälkeen tehdään harmaasävykuvasta mustavalkoinen mukautuvalla kynällä.



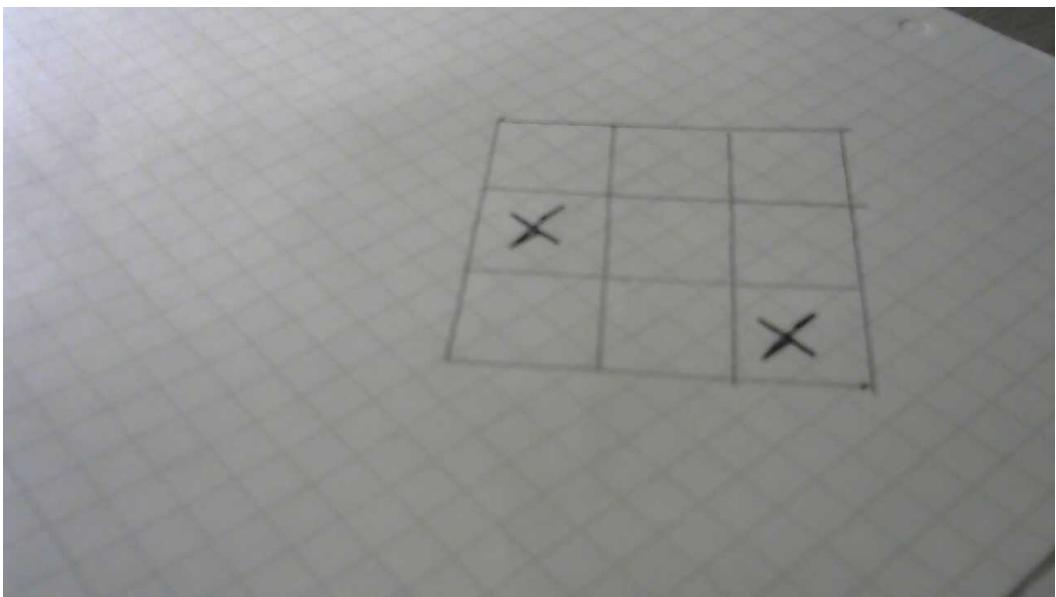
Kuva 10: Kameran asettelu, edestä

nystyksellä (suomennos??, siis [adaptive threshold](#)), jolloin kuvaan jää jäljelle vain pääasiassa merkitseviä viivoja ja merkkejä. Tämän 'binäärikuvan' värit käännetään jatkoa varten.

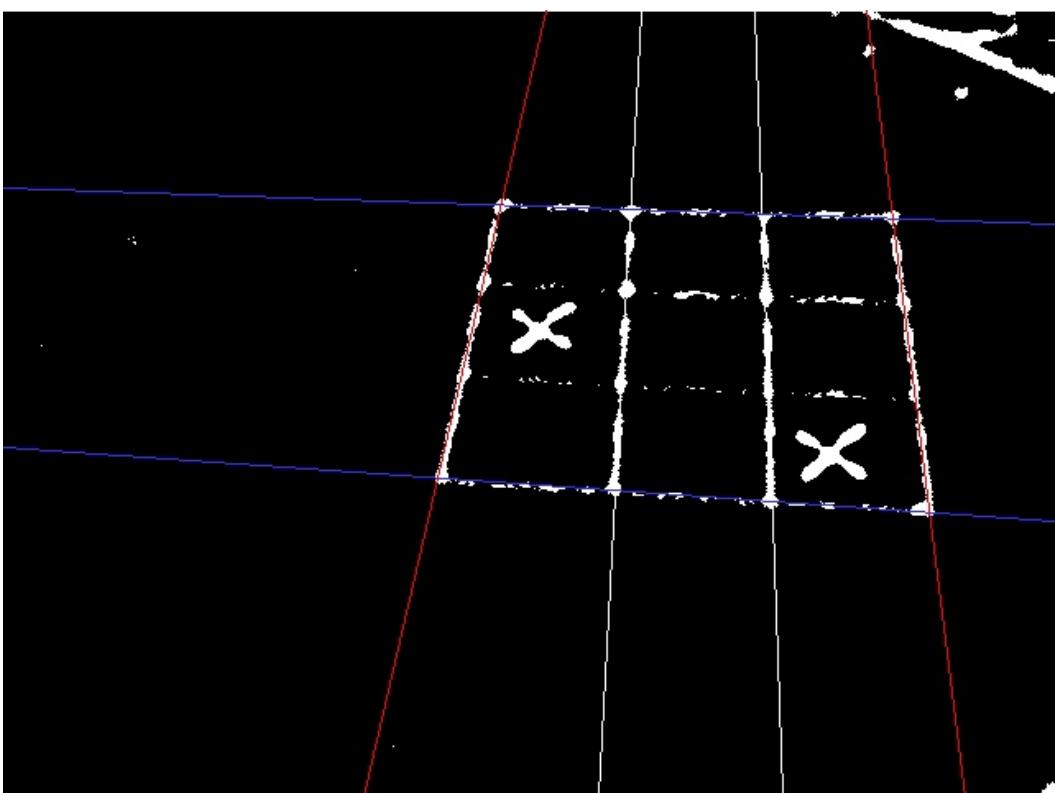
4. Aiemmassa vaiheessa jotkut tärkeätkin ruudukon viivat saattavat 'katketa', jojen niitä yritetään palauttaa morphologisella sulkemisella ([morphological closing](#)).
5. Näin käsitellystä kuvasta etsitään [Hough-muunnoksella](#) kaikki viivat.
6. Koska OpenCV:n Hough-rutiini löytää sellaisilla parametreilla, joilla varmasti saadaan kaikki *tärkeät* viivat, myös *paljon* ylimääräisiä viivoja jokaista pelilaudan todellista viivaa kohti, lähellä toisiaan olevien viivojen parvet yhdistetään yhdeksi viivaksi per parvi (keskiarvo).
7. Yhdistetyistä viivoista etsitään äärimmäiset (tietyn marginaalin puitteissa) vaaka- ja pystyviivat, ts ääriviivat jotka vastaavat pelilaudan reunoja. Näiden leikkauspisteet (= peliruudukon nurkat) lasketaan.
8. Leikkauspisteiden avulla kuvan perspektiivi korjataan ja se jaetaan 3x3 -ruudukoksi. Kunkin ruudun reunoista 'leikataan pois' pieni kaistale (jotka sisältävät piirretyn ruudukon viivat) ja sen sisäalue ja sisäalueen histogrammi talletetaan.
9. Jokaiselle verrattavalle kuvalle tehdään sama prosessi, ja kuvien vastaavia

alueita verrataan toisiinsa. Mikäli jonkin solun histogrammeissa peruskuvan ja verrattavan välillä on suuri ero, todetaan että tähän ruutuun on verrattavassa kuvassa piirretty uusi merkki.

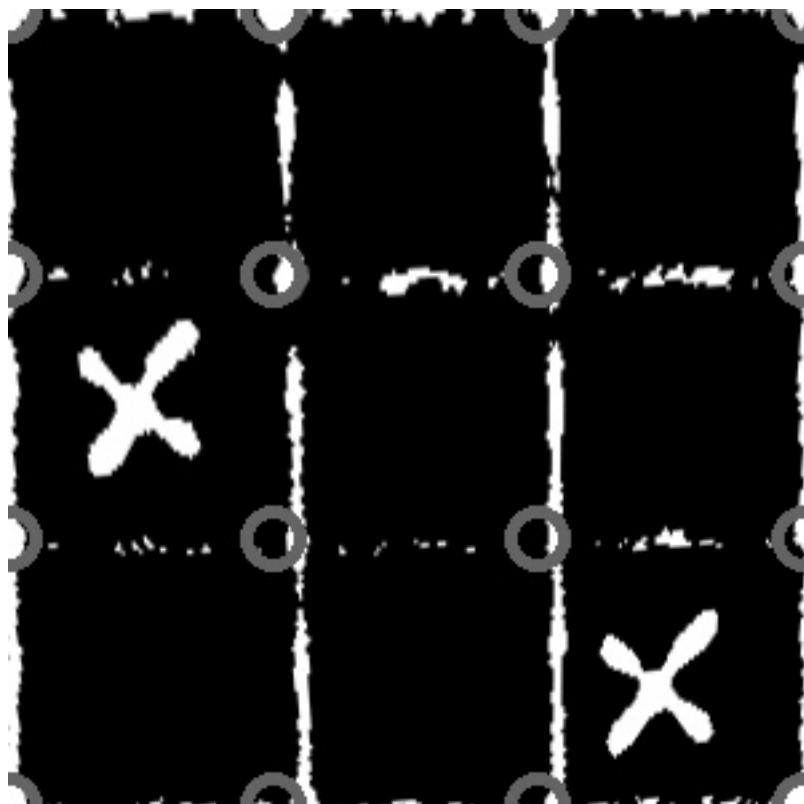
10. Mikäli havaittu merkki hyväksytään oikein luetuksi siirroksi, se päivitetään uudeksi peruskuvaksi seuraavan siirron lukemista varten.



Kuva 11: Kameran näkymä



Kuva 12: Kameran näkymästä tunnistetut pelilauden ääriviivat



Kuva 13: Perspektiivikorjaus ja ruudukon hahmottaminen

3 Ohjelmakoodi

Koodia mielestäni kommentoitu runsaasti ja sillä on JavaDoc (kunhan keksin minne uploadaan ne), joten tässä esittelen ohjelmakoodia melko yleisellä tasolla.

3.1 PenBot

PenBot:in ohjelmakoodin rakenne on seuraava:

- `comms`
 - `Command.java` bluetooth-kommunikaatiokomennot
- `penbot` varsinainen piirtotoiminnallisus
 - `Penbot.java` robotin päätoiminnot: piirturin liikkuminen, kynän ohjaus; tarjoaa julkiset metodit merkin piirtämiseksi tiettyyn koordinaattiin
 - `Board.java` pelilaudan sisäinen malli: tarjoaa `Penbot.java:n` käyttämät liikkumisohjeet kuhunkin pelilaudan ruudukkoon
- `main` päätölkäytäntö
 - `BotControl.java` piirturin käskyttäminen LeJOS:n Bluetooth-yhteyden tarjoaman `InputStream`:n kautta saapuvien komentojen mukaisesti
 - `Main.java` Ohjelman käynnistys (Bluetooth-yhteyden avaus) ja lopetus, häitäseis `ButtonListener:n` asettaminen
- `test`
 - `PenConfigureTest` testiohjelma kynämoottorin säättämiseen

3.1.1 Huomioita

- pelikentän ulottuvuuksia (ruutujen koko, botin etäisyys ruudukosta, jne) voi toistaiseksi säättää vain asettamalla vastaavan `Penbot.java:n` vakiot sopiviksi.
- bluetooth-`InputStream`ista luetaan käskyjä vain piirtosuoritusten välissä: ensimmäistä vastaanotettua käskyä aletaan toteuttaa välittömästi
- hätipysäytys toteutettu `ButtonListenerillä`

3.2 BotGame

- `comms` viestintäpätkä
 - `Command.java` bluetooth-kommunikaatiokomennot
 - `BTComms.java` abstraktio komentojen lähetämiseksi bluetoothin yli

- `compvision` keinonäköpaketti
 - `BoardReader.java` varsinainen pelilaudan hahmotus ('lukeminen' ku vasta)
 - `Grid.java` peliruudikon sisäinen malli ja vertailumetodit
 - `Morphs.java` morphologisia operaatiota
 - `Webcam.java` web-kameran kuvan käsittely
 - `compvision.lineutils` lineaarialgebran työkaluja:
 - * `Lines.java` staattisia metodeja viivojen käsittelyyn (tulostus, leik kauskohtien laskeminen)
 - * `LineGroups.java` viivaryhmäolio
- `game` varsinainen peli
 - `BotControl.java` (*huono nimi ja hieman turha*) PenBot:in `BotControl`:n vastinpari, AI:n siirtojen suorittaminen `BTComms`:n avulla
 - `Game.java` varsinainen peliloop
 - `Player.java` interface, jonka kaikki erilaiset 'pelaajat' (AI:t tai käyttöliit tymä pelaajat) toteuttavat, jotta `Game.java` osaa viestiä eri niiden kanssa
 - `game.ui` käyttäjäinteraktiopaketti
 - * `StandardPlayer.java` ristinollan pelaaminen 'pelkkänä' konsolissa ilman web-kameraa (käyttäjä antaa käsin koordinaatit siirroilleen)
 - * `AssistedWebcamPlayer.java` laajennettu `StandardPlayer`: pelaajan liikkeiden lukeminen web-kamerasta `compvision`:n avulla ja luke mien vahvistusten pyytäminen (tätä käytetään)
 - `game.ai` pelitekoälypaketti
 - * `SimpleBotAi.java` toistaiseksi ainoa tekoäly; täytyää ruudukosta seu ravaan vapaan ruudun omalla merkillään
 - `game.domain` peliobjektien mallinnus
 - * `Board.java` pelilaudan tilanne
 - * `GameMove.java` pelisiirtoa kuvaava olio (X vai O, koordinaatti)
 - * `Mark.java` pelimerkkiä (X, O) kuvaava enum
- `main`
 - `Main.java` ohjelman käynnistys, lopetus
- `test` sisältää testiohjelman `compvision`:n kokeiluun

3.2.1 Huomioita

- Webcam.java ajaa web-kameran kuvasyötteen lukua omassa säikeessään, jossa pyytää kameralta uusia frameja sitä mukaa kuin kamera niitä osaa antaa ja pitää uusinta ‘puskurissa’. Webcam.java tarjoaa julkisen metodin, jolla saadaan kopio ko. muistiin säilötystä framesta jatkokäyttöä varten. Säikeistys oli minulle uusi asia, mutta kopioinnin pitäisi toimia ‘säieturvallisesti’ (*thread safe*): Webcamin sisäinen frame -puskuri lukitaan synchronized-metodeilla kopioinnin ajaksi.
- Kuvanlukua käytetään hieman typerästi vain AssistedWebcamPlayer:stä käsin, jonka vuoksi mm. botti-AI:n omien siirtojen aiheuttamia muutoksia ei voi suoraan ohittaa. Jälkkäteen järkevämpi tapa olisi ollut että keinonäköä olisi käyttänyt peli itse (Game.java), joka olisi tarjonneet pelaajille mahdollisuuden saada tietoa keinonäön käsityksestä pelitilanteesta tms. Syy nykyiseen huonoon ratkaisuun on että kirjoitin nopeasti Assisted:n StandardPlayer:n päälle saadakseni nopeasti joitain toimivaa demotilaisuutta varten.
- Keinonäön matemaattista toimintaa on selostettu yleisellä tasolla edellisissä kappaleissa.

4 Testaus

Kaikkea mitä olisi voinut testata, ei tullut testattua. Erityisesti matematiikkameteoja ja kuvanhahmotustoimintaa varten olisi voinut kirjoittaa suoranaisia yksikkö-testejä esim. JUnitilla.

Käytännössä robotin kehittäminen oli iteratiivinen prosessi: “testataan toimiiko jokin toiminnallisuus näin” -> “korjataan kunnes toimii” -> “kun toimii, lisätään toiminnallisuus”. Valitettavasti tälläisestä epä-TDD ‘patternista’ ei jänyt hirveästi varsinaista testikoodia.

4.1 Ohjelmallisia testi’skriptejä’

Eri toiminnallisuksien kokeilemista ja säättöä varten on ohjelmissa erityisen test-paketin luokissa muutamia main-metodeja, joita ajamalla varsinaisen Main.main:n sijaan voi testata ko. luokkien toimintaa.

4.1.1 PenBotin kynänliikuttelun testaus ja säätö

PenBot:n test.PenConfigureTest sisältää testin kynän kalibointirutiinille, jonka avulla voi kokeilla yleisesti kynänliikuttelun toimivuutta (“liikkuuko kynä oikein? piirtääkö se jäljen paperille?”).

Lisäksi PenConfigureTest mahdollisti piirtobotin kynänliikuttelun ohjelmallisten turvarajojen testaamisen. Tulos: turvarajat toimivat, kalibrointiskriptin (= normaali käyttö) avulla bottia ei saatu käänämään kynämoottoria yli 20 astetta alas-päin, joka oli todettu vielä täysin turvalliseksi asennoksi.

4.1.2 Kuvanhahmotuksen testaus ja säätö

BotGame:n `test.BoardReaderCamTest`:ia voi käyttää web-kameran kuvankaappauden toiminnan testaamiseen (“havaitseko kuvantunnistusmenetelmä ruudukkoon piirretyn uuden X- tai O-merkin oikein?”) käynnistämättä varsinaista pelirutiinia. Esimerkiksi tarkistin ennen demotilaisuutta että kuvankäsittelymetodit toimivat myös yliopiston tiloissa (jossa olisi voinut olla esim. eri valaistusolot kuin kotona).

Vastaavankaltaista koodinpätkää käytettiin merkintunnistustoiminnallisuuutta koodatessa myös eri raja-arvojen ja kuvaruutujen vertailumenetelmien tutkimiseen. Lopulta päädyttiin koodin tämänhetkisessä versiossa oleviin vakioihin ja metodeihin.

4.1.3 Käskyjen välittäminen tietokoneelta PenBotille ja piirtäminen

Bluetooth-kommunikaation testaamista varten on erillinen komentoriviohjelma `BotCommander`, jonka avulla käyttäjä voi suoraan komentoriviltä käskyttämällä lähettää viestintäprotokollan mukaisia komentoja `PenBot`:lle.

`BotCommander`-in avulla tehtiin seuraavat testit:

1. Bluetooth-yhteyden muodostaminen ja `Input/OutputStream`:n avaaminen `PenBot`in ja `BotCommander`-in välillä onnistuu.
2. `PenBot` vastaanottaa ja lukee Bluetoothin kautta lähetettyjä käskybittejä onnistuneesti.
3. `PenBot` suorittaa käskyn mukaisen komennon oikein (piirtää ruksin oikeaan koordinaattiin).

4.2 Testibotit

Varsinaisen `PenBot`-ohjelman lisäksi jäljelle jäi pari pieniä ‘testibottia’ jotka voitiin myös ladata Lego-robotin brickille LeJOS:n yms. eri ominaisuuksien testaamiseksi.

4.2.1 Hello Ironman!

Käytettiin testaamaan toimiiko yksinkertaisen “hello world” -ohjelman käänämisen ja lataaminen robottiin eri ympäristöissä ja yhteysmenetelmissä. Jouduttiin mm. toteamaan että Ubuntu ajureilla ei saanut toimivaa USB-yhteyttä Lego-robottiin.

4.2.2 Hello BT!

Testattiin viestibittien vastaanottamisen lisäksi myös lähetämistä robotilta tietokoneelle, mutta tästä ominaisuutta ei sitten varsinaisessa pelirobotin toteutuksessa hyödynnetty.

4.3 Muita testiskenaarioita

4.3.1 Hätipysäytys

PenBot:iin asetettua vaadittua hatätipysäytystoiminnallisuutta testattiin painamalla kesken ohjelman suorituksen pysäytysnapiksi valittua ESCAPE-nappulaa. Hätipysäytys toimi. [Video testistä](#)

5 Rajoitukset ja tulevaisuus

5.1 Toteuttamatta jäneet ominaisuudet

Robotti jää kahdelta osin hieman keskeneräiseksi:

Ensinnäkin, peli ei osaa pelata ristinollaa täysin itsenäisesti, sillä BotGame:n hahmontunnustuskoodi ei osaa hylätä sellaisia webkameran kuvia, joissa pelilauden ja kameran välissä on este (esimerkiksi ihmispelaajan käsi tai piirtorobotti itse). Ohjelmasta puuttuu myös botin piirtämien merkkien tunnistus kameran kuvasta AI:n omaksi siirroiksi. Tämän vuoksi peli pyytää käyttäjältä vahvistuksen jokaiselle pelilaudalla havaitulle muutokselle esittääkö se botin tai pelaajan tekemää siirtoa.

Toiseksi varsinainen tekoäly puuttuu. Robotti pelaa ristinollaa säätöjen mukaan, muttei erityisen älykkäästi.

Molemmat ongelmat olisi ollut tarkoitus ratkaista: Ristinollatekoälyn toteutus jonkinlaisella minimax-algoritmilla tai alpha-beta -karsinnalla olisi melko triviaali tehtävä. Kameran eteen tulleen esteen kaltaiset huomattavat muutokset kuvassa puolestaan olisi (ainakin teoriassa) yksinkertaista havaita OpenCV:n avulla.

Esimerkiksi eräs vaihtoehto tähän olisi tarkastella värikuvan histogrammin poikkeamia (pelilautaa esittävään kuvaan nähdyn) kun laudan päällä on robotin tai käden kaltainen 'ylimääräinen' esine. Oletettavasti histogrammissa nähtäisiin suuri poikkeama verrattuna tilanteeseen, jossa ainoa muutos on peliruutuun ilmestynyt pieni merkki.

Botti-tekoälyn tekemät siirrot vuorostaan olisi luultavasti mahdollista tunnistaa (ja ohittaa kysymättä pelaajan vahvistusta) hieman ohjelmakoodia laajentamalla.

5.2 Muita puutteita ja rajoitteita:

Piirtorobotti ei osaa asemoida itseään pelilautaan nähdien. Käyttäjän on sijoitettava robotti ennaltavalittuun asentoon peliruudukkoon nähdien (pelilauden lävistäjän kautta kulkevalle suoralle). Mikäli robotti on hieman vinossa, se myös ajaa hieman sivuun ja pahimmassa tapauksessa piirtää ristejä väriin paikkoihin. Ongelmaa voi siinä hieman helpottaa laajentamalla nykyistä toiminnallisuutta pienellä kalibointiskriptillä (robotti kulkisi edestakaisin ja piirtäisi pisteytä sinne missä se kuvittelee esim. peliruudukan nurkkien olevan; käyttäjä voisi korjata robotin asentoa).

Piirtorobotti nykyisessä muodossaan piirtää kulkemalla edestakaisin ja pyörimällä moottoriakselinsa keskipisteen suhteen; toisin sanoen vaakasuuntainen viiva on kaareva. Tämän vuoksi mahdolliset kuviot käytännössä ovat ruksien ja pisteen kaltaisia yksinkertaisia kuvioita, joita tämä rajoitus ei haittaa. Ristinollan perinteisen 'nolla'-kuvion piirtäminen olisi nykyisellä rakenteella melko vaikeaa.

Kuvan analysointimenetelmät ovat teoriatasolla yleistettävissä, mutta koodissa käytetyt vakiot, raja-arvot, jne. on löydetty käsin kokeilemalla tietynlaisella kameralokeroonpanolla. Esimerkiksi huomasin että tutkittavien kuvien resoluution vaihtaminen voi rikkoa nykyisen toiminnallisuuden.

6 Käyttöohjeet

6.1 Ohjelmistojen käääntäminen ja lataaminen NXT:lle

6.1.1 PenBot

Tarvitaan:

- LeJOS 0.9.1 (omine vaatimuksineen, ts. tuore JDK); LeJOS täytyy kirjoittaa Legon oman firmwareen päälle NXT-brickille (LeJOS:in omien ohjeiden mukaan).
- Uusin Ant.
- Toimiva Bluetooth- tai USB-yhteys käääntöympäristön ja NXT:n välillä.

Koodin mukana tulee LeJOS:in käyttöesimerkkien mukainen Ant-skripti. Aja esim. `ant uploadandrun` kansiossa PenBot.

Kääntäminen toimii ainakin Ubuntu 14.10:llä, mutta todennäköisesti onnistuu myös Windows / Mac OS. (Ei kokeiltu.)

6.1.2 BotGame

- Edellisten lisäksi LeJOS tarvitsemat OS-natiivit Bluetooth-kirjastot (seuraa LeJOS:in asennusohjeita).

- OpenCV 2.4.4+ (toimii ainakin 2.4.9.0.) vaatimuksineen. Käännä OpenCV:n kirjastot asennusohjeiden mukaisesti.

BotGamea varten ei ole kätevää Ant-skriptiä; sen sijaan GitHubissa on valmis Eclipse-projekti, johon täytyy linkata vaaditut LeJOS- ja OpenCV-kirjastot.

6.2 Pelaaminen

Pelitoiminnallisuus on edelleen hieman raakile.

Olettaen että PenBot:n lähdekoodiin on kirjoitettu oikeat pelialueen ja robotin mitat (robotin etäisyys ruudukosta, ruutujen koko, robotin mitat):

1. Aseta PenBot oikealle etäisyydelle (säädä lähdekoodia) ruudukosta, tarkalleen ruudukon lävistäjän kautta kulkevalle suoralle (ks kuvat), ja kohdista web-kamera.
2. Käynnistä PenBot NXT:llä. Näytölle ilmestyy teksti I AM IRONMAN. Paina mitä tahansa näppäintä paitsi harmaa ESCAPE (joka kaikissa tilanteissa keskeyttää ohjelman toiminnan).
3. PenBot siirtyy kynän kalibrointilaan. Säädä kynämoottorin käantökulma nuolinäppäimillä niin että kynä piirtää selkeän jäljen paperille, ja paina oranssi ENTER.
4. Näytölle ilmestyy teksti WAITING FOR CONNECTION. Käynnistä BotGame (tai BotCommander).
5. Kun yhteys on luotu, BotGame tulostaa stdout:iin Connected! ja pyytää tarkistamaan webkameran asennon. Tätä varten aukeaa ikkuna web-kameran näkymästä. Kun kamera on kohdallaan, paina rivinvaihtoa. Tämän jälkeen BotGame yrittää etsiä kameran kuvasta ruudukon, ja onnistuessaan tulostaa lyötämänsä ääriviivat.
6. PenBot puolestaan ilmoittaa että se on valmis kommunikoimaan BotGame:n kanssa viestillä READY TO READ: paina taas jotain NXT:n näppäintä.
7. Paina rivinvaihtoa; BotGame aloittaa pelin.
8. Konsoliin tulostuu BotGame:n tekemä siirto ja PenBot piirtää sen. Mikäli kamera toimii, se pienentää viiveen jälkeen (jonka aikana robotti ehtii toivottavasti ajaa pois kameran edestä) havaitsee laudalla muutoksen ja kysyy onko se robotti vai pelajaan merkki (vai virheellinen havainto). Vastaa bot jos havaittu siirto on oikea (tarkista suorituskansioon tulostetut debug-kuvat) tai no jos robotti ei ehtinyt alta pois.
9. Tee oma siirtosi. Merkin havaittuaan BotGame:n pitäisi jälleen kysyä kenen se on; vastaa jälleen. BotGame tulostaa konsoliin missä se mielestään näki symbolisi.

10. Robotti vastaa omalla siirrollaan. Jatketaan samaan tapaan, kunnes jompi kumpi on voittanut pelin (jolloin robotti ilmoitta kumpi voitti, ja lopettaa).