

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	10
3.1 Penbot	10
3.2 BotGame	10
4 Testaus	10
4.1 Ohjelmallisia testi'skriptejä'	11
4.1.1 PenBotin kynänliikuttelun testaus ja säätö	11
4.1.2 Kuvananalysointitoiminnallisuuden testaus ja säätö	11
4.1.3 Käskyjen välittäminen tietokoneelta PenBotille ja piirtäminen	12
4.2 Testibotit	13
4.2.1 Hello Ironman!	13
4.2.2 Hello BT!	13
4.3 Muita testiskenaarioita	13
4.3.1 Hätipysäytys	13
5 Rajoitukset ja tulevaisuus	13
5.1 Toteuttamatta jääneet ominaisuudet	13
5.2 Muita puutteita ja rajoitteita:	14
6 Käyttöohjeet	14
6.1 Ohjelmistojen käänтäminen	14
6.1.1 Penbot	14
6.1.2 BotGame	14
6.2 Pelaaminen	14

Tämä on joulun 2014 robottikurssin 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 piirretyn pelilauden ja pelaajien (robotti ja sen vastustaja) sille piirtämät merkit, ja tämän avulla pelata ristinollaan ihmisvastustajaa vastaan. Peliohjelma lähetää tekoälyn valitsemat siirrot Bluetoothin yli piirtorobottiin, jolla on valmiit ruttiinit ruksin piirtämiseksi 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

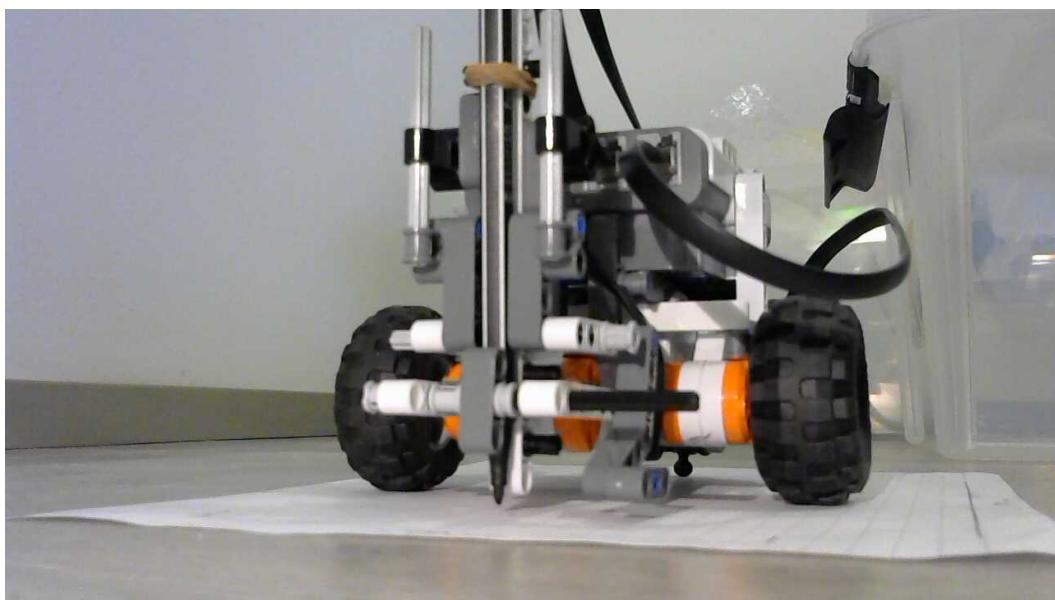
heilumista rajoittavat tuet, jotka pitävät kynän tukevasti paikoillaan piirtämisen aikana.

[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.

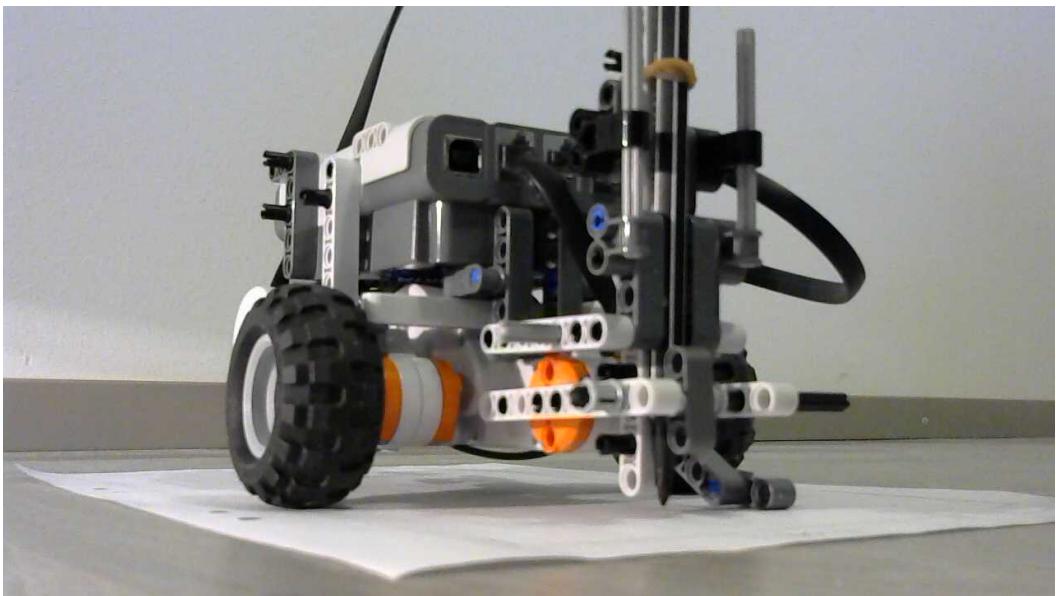
2.2.3 Ruksin piirtäminen

Robotti osaa piirtää pelilaudalle ruksin edestakaisella liikkeellä ja sivusuuntaisilla käänöksillä.

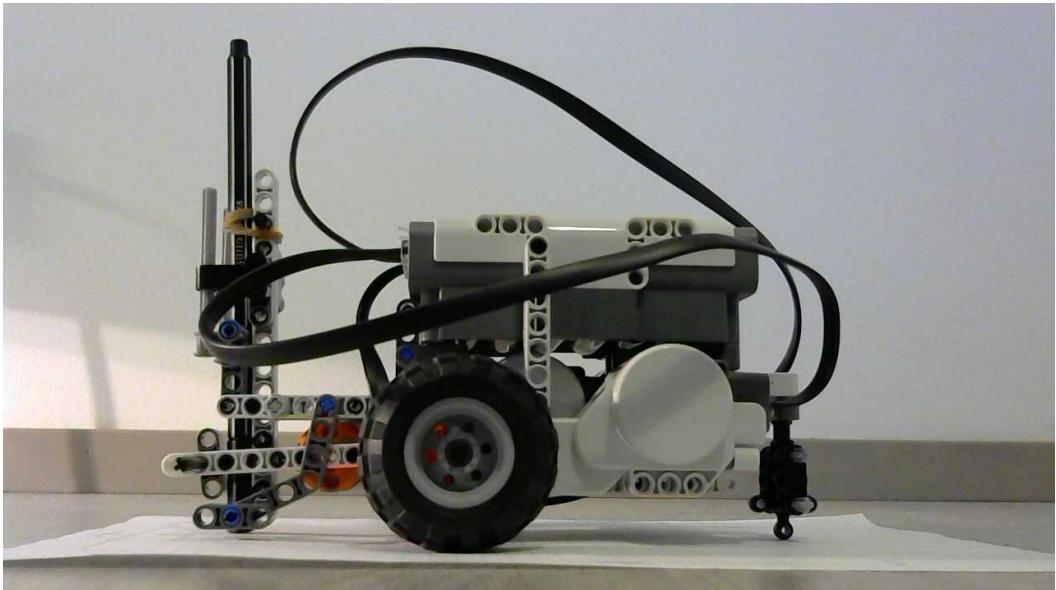
Robotti piirtää yhden ruksin.

Robotti piirtää toisen ruksin.

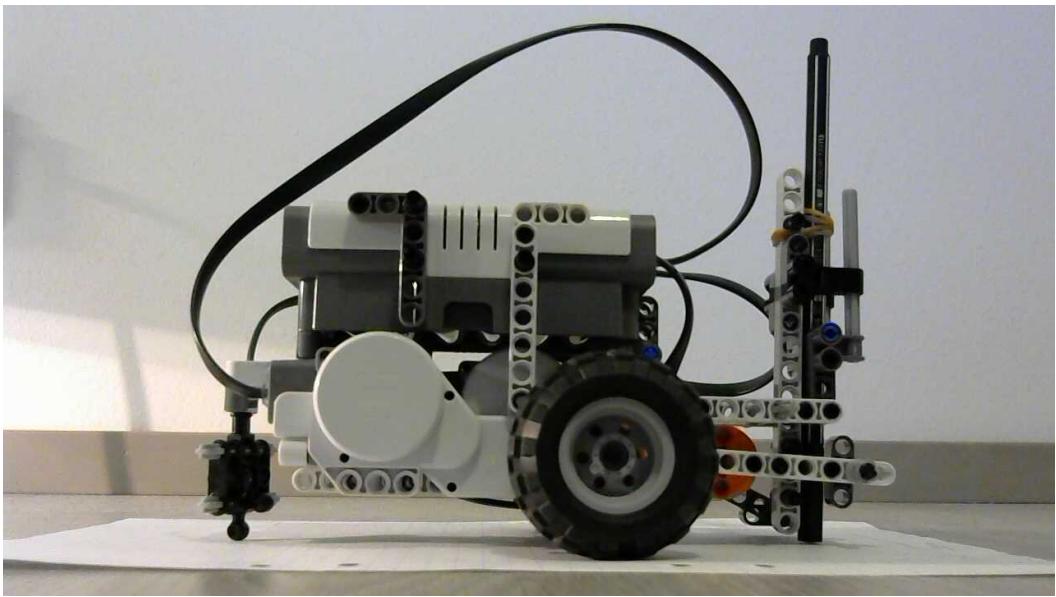
Väärin suunnattu robotti piirtää ruksin hieman sivuun ruudun keskipisteestä.



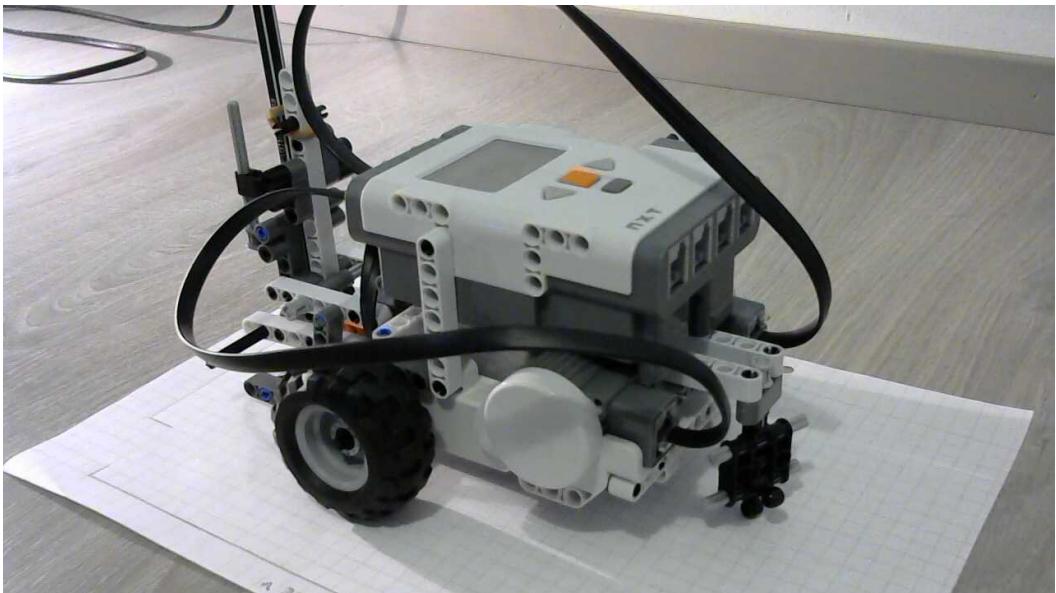
Kuva 3: Toinen kuva edestä (kynämekanismi)



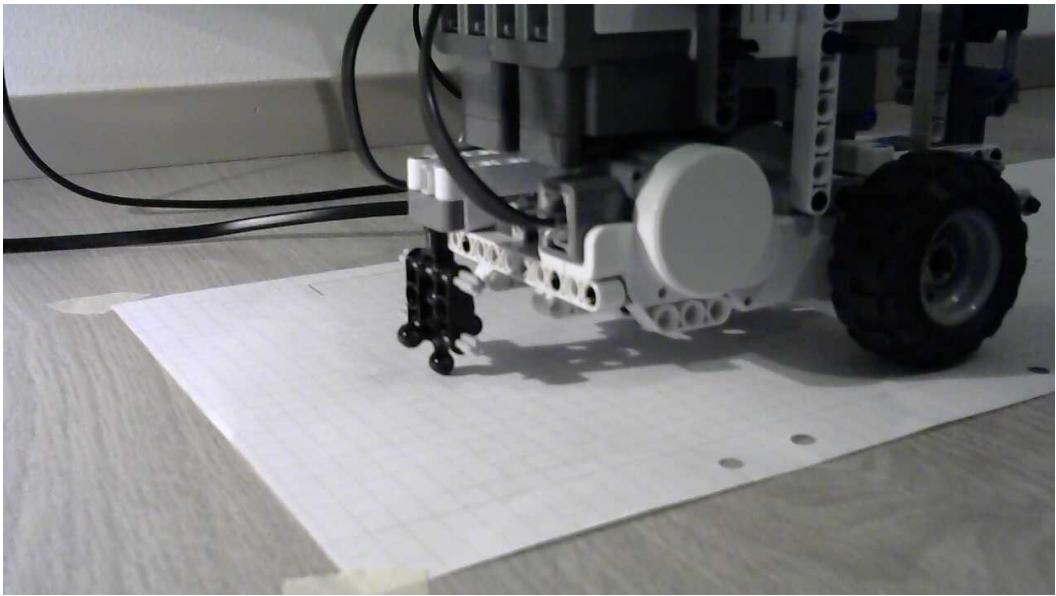
Kuva 4: Sivusta.



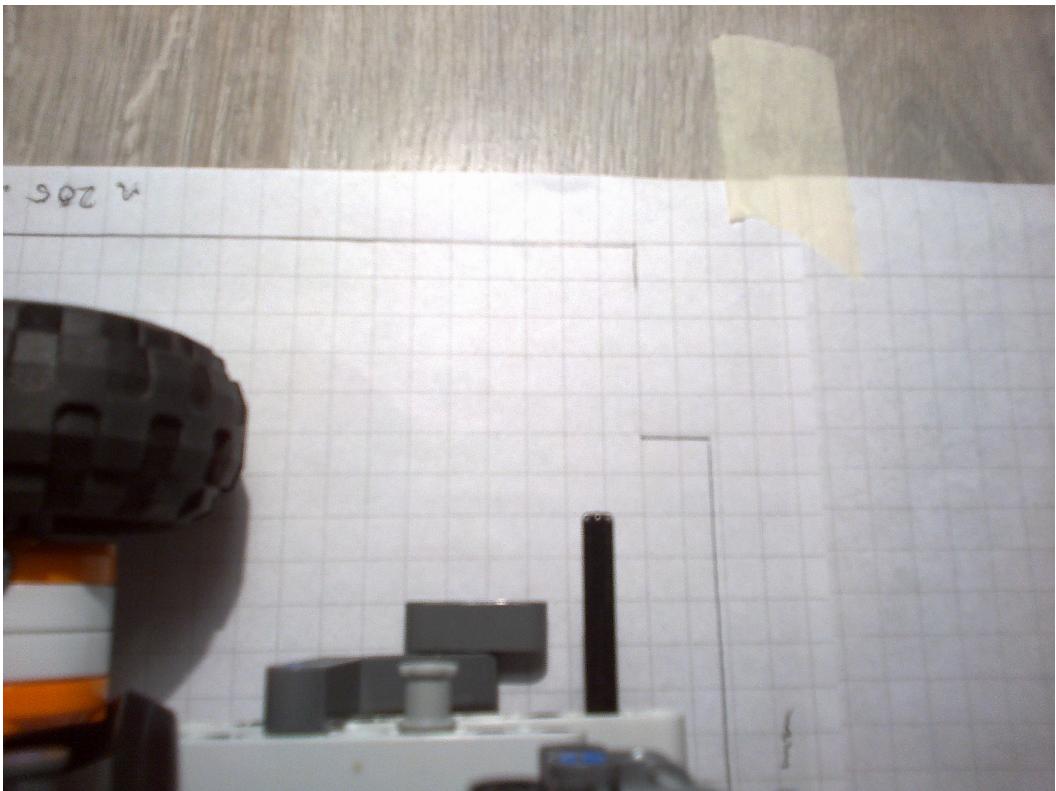
Kuva 5: Toiselta sivulta.



Kuva 6: Takaan.



Kuva 7: Peräosa

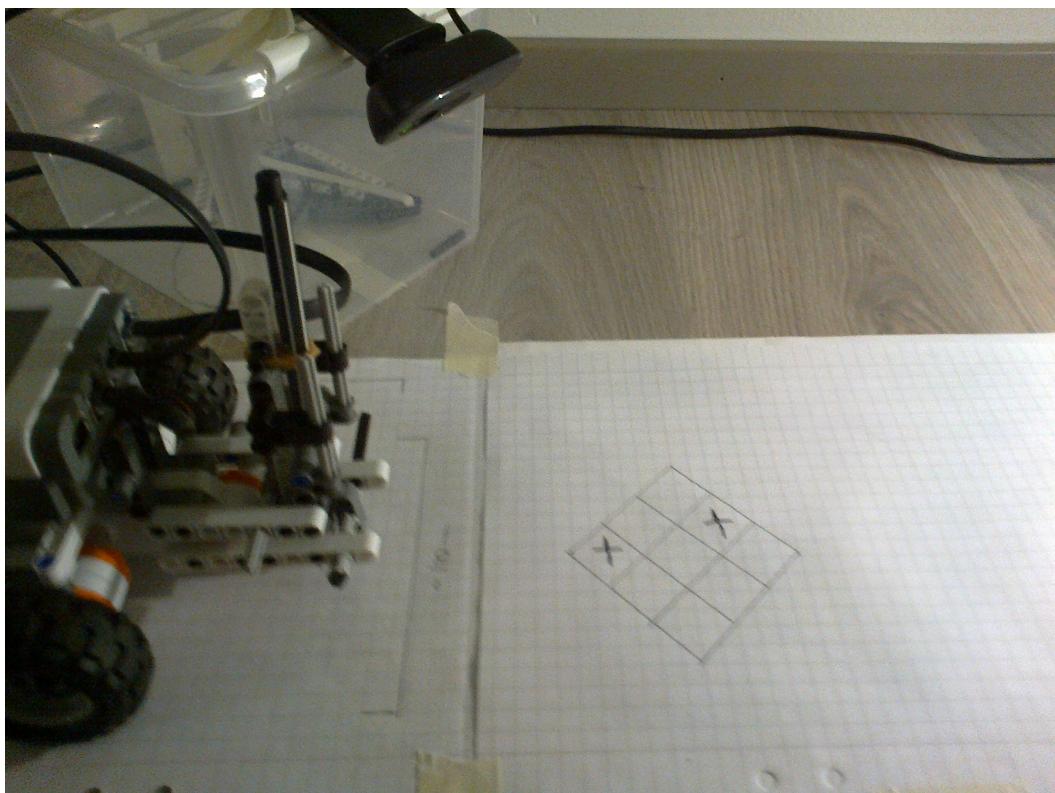


Kuva 8: Robotin asennon tarkistaminen.

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 muita mahdol lisesti pysty- tai vaakasuuntaisilta viivoilta näyttäviä kohteita (paperin reuna, varjot, jne) jotka saattavat erehdyttää BotGame:n pelialueen hahmotusmetodeja.

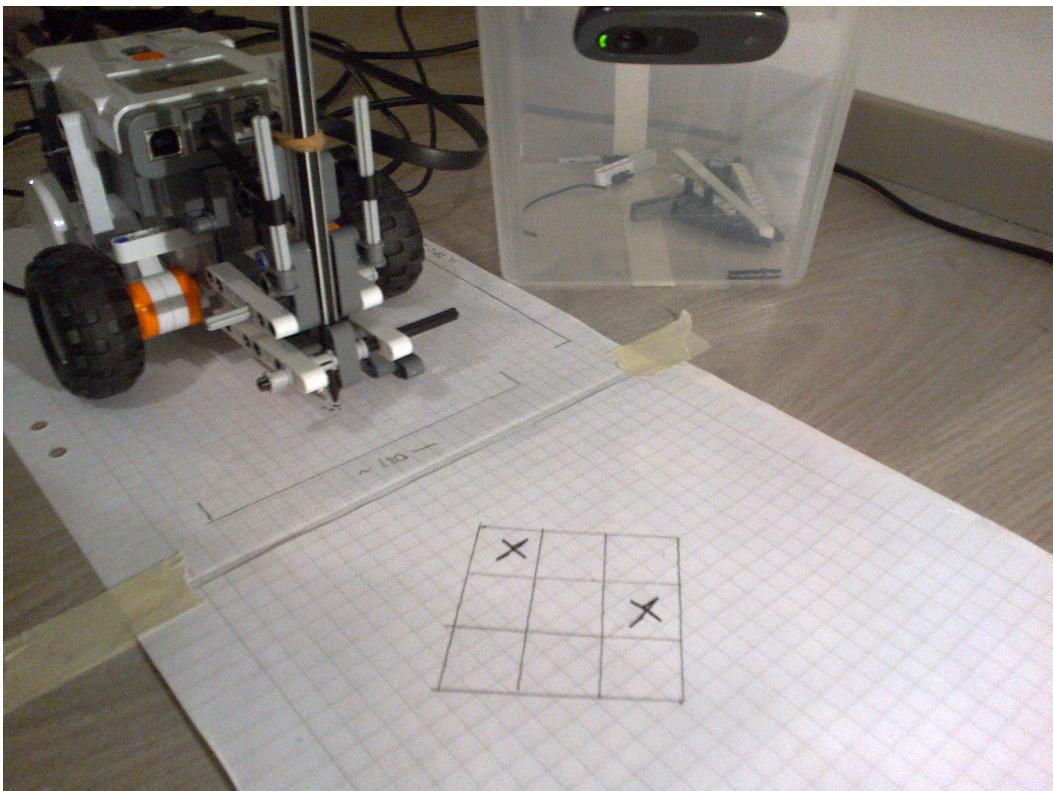


Kuva 9: Kameran asettelu, sivukuva

2.3.2 Kuvantunnistuksen toimintaidea

Kuvantunnistusmenetelmän pääinspiraationa oli [AI Shackin Sudoku-lukija](#), jota tosin on sovellettu varsin 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 muuttamalla ([adap-](#)



Kuva 10: Kameran asettelu, edestä

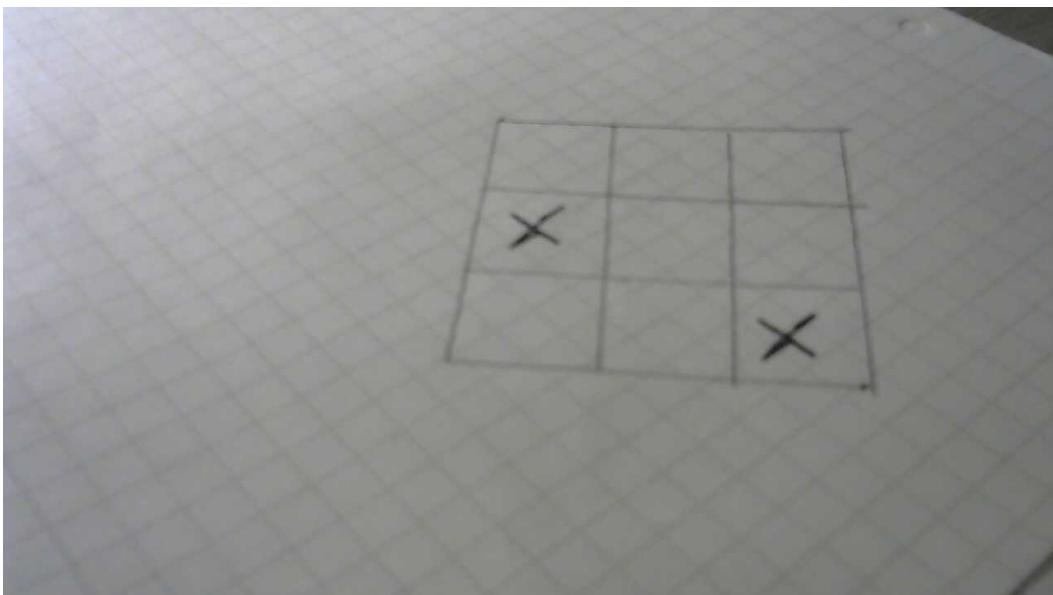
tive threshold), jolloin kuvaan jää jäljelle vain pääasiassa merkitseviä viivoja ja merkkejä. Tämän 'binäärikuvan' värit käännetään jatkoaa varten.

4. Aiempassa vaiheessa jotkut tärkeätkin ruudukon viivat saattavat 'katketa', joten niitä yritetään palauttaa morphologisella sulkemisella (**morphological closing**).
5. Näin käsittelystä kuvasta etsitään **Hough-muunnoksella** kaikki viivat.
6. Koska OpenCV:n Hough-rutiini löytää sellaisilla parametreilla joilla varmasti saadaan kaikki *tärkeät* viivat myös *paljon* viivoja jokaista pelilaudan oikeaa viivaa kohti, lähellä toisiaan olevien viivojen parvet yhdistetään yhdeksi viivaksi per parvi (keskiarvo).
7. Yhdistetyistä viivoista etsitään äärimmäiset (tietyn marginaalin puitteissa) vaaka- ja pystyviivat, 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 (alkutilanteessa tyhjä) sisäalue ja sen histogrammi talletetaan.
9. Jokaiselle verrattavalle kuvalle tehdään sama prosessi, ja kuvien vastaavia

alueille 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.

TODO Kuvia laudan hahmottamisesta.



Kuva 11: Kameran näkymä

3 Ohjelmakoodi

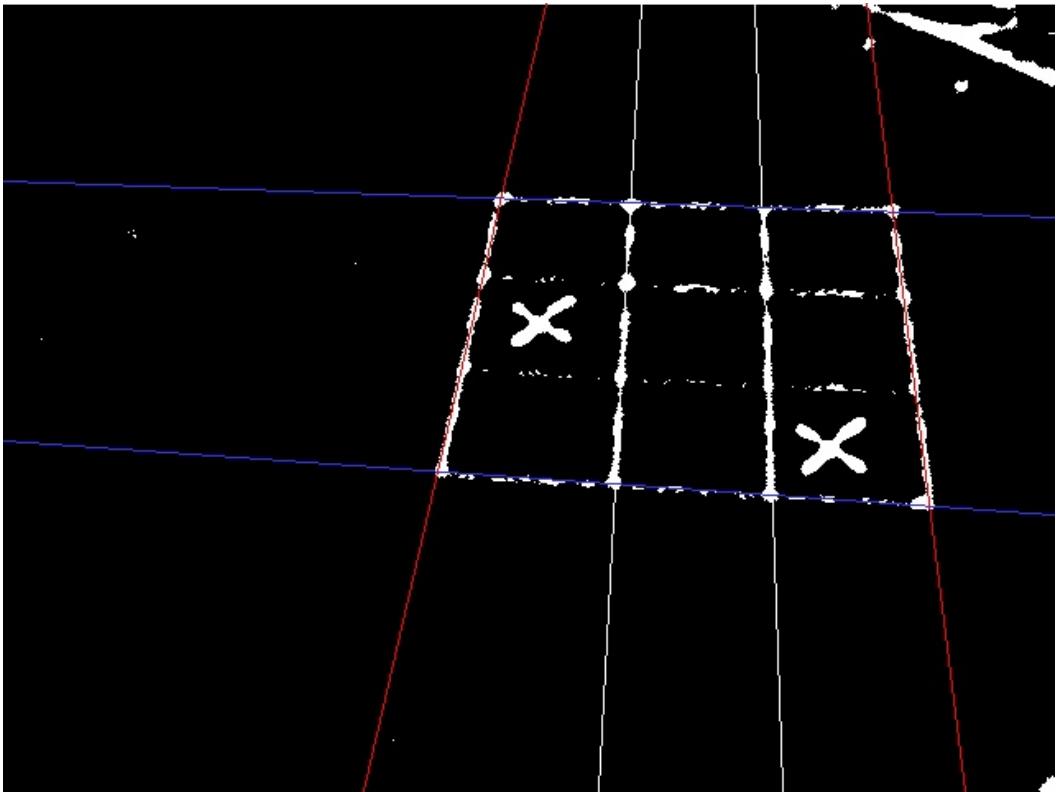
3.1 Penbot

3.2 BotGame

4 Testaus

Kaikkea mitä olisi voinut testata, ei tullut testattua. Erityisesti matematiikkametodeja ja kuvanhahmotustoimintaa varten olisi voinut kirjoittaa suoranaisia yksikkötestejä.

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.



Kuva 12: Kameran näkymästä tunnistetut pelilaudan ääriiviivat

4.1 Ohjelmallisia testi'skriptejä'

Eri toiminnallisuksien kokeilemista ja säätöä varten on ohjelmissa erityisen testipaketin luokissa muutama `main`-metodeja, joita ajamalla varsinaisen `main`:n sijaan voi testata robotin eri toiminnallisuksien toimintaa.

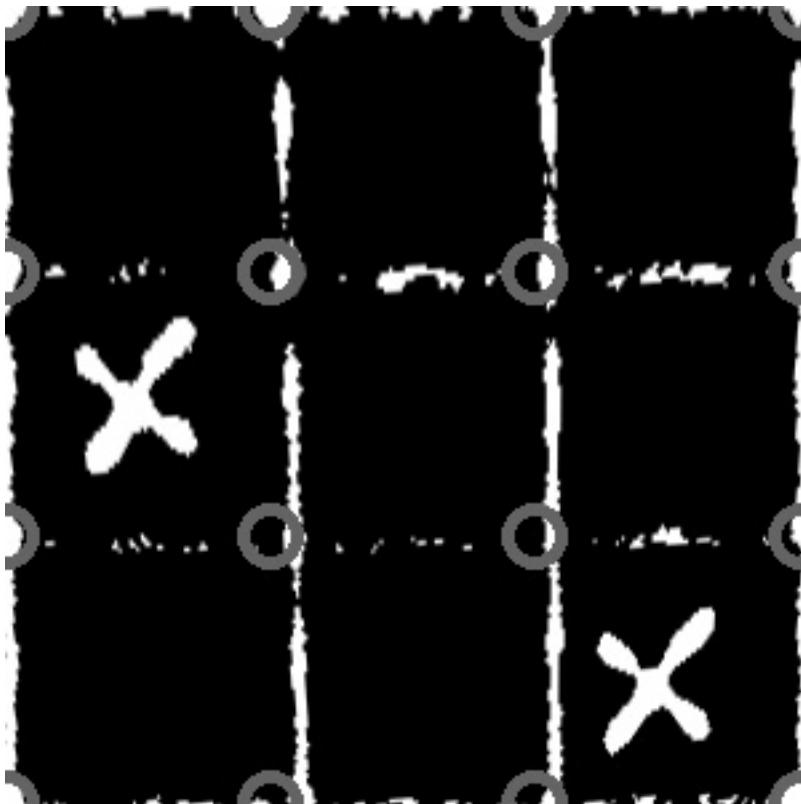
4.1.1 PenBotin kynänliikuttelun testaus ja säätö

`PenBot`:n `test.PenConfigureTest` sisältää testin kynän kalibrointirutiinille, 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 kääntämään kynämoottoria yli 20 astetta alas-päin, joka oli todettu vielä täysin turvalliseksi asennoksi.

4.1.2 Kuvanalysointitoiminnallisuuden testaus ja säätö

`BotGame`:n `test.BoardReaderCamTest`:ia voi käyttää web-kameran kuvankaappausen toiminnan testaamiseen käynnistämättä varsinaista pelirutiinia. Esimerkiksi



Kuva 13: Perspektiivikorjaus ja ruudukon hahmottaminen

tarkistin ennen demotilaisuutta että kuvankäsittelymetodit toimivat myös yliopiston tilojen valaistuksessa.

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 lähettilä viestintäprotokollan mukaisia komentoja PenBot:ille.

`BotCommander`-in avulla tehtiin seuraavat testit:

1. Bluetooth-yhteyden muodostaminen ja `Input/OutputStream:n` avaaminen PenBotin 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 pientä ‘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 kääntäminnen ja lataaminen robottiin eri ympäristöissä ja yhteysmenetelmillä. Jouduttiin mm. toteamaan että Ubuntun 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 Hätipysäyts

PenBot:iin asetettua vaadittua hatäpysäytystoiminnallisuutta testattiin painamalla kesken ohjelman suorituksen pysäytysnapiksi valittua ESCAPE-nappulaa. Hätipysäyts toimi.

5 Rajoitukset ja tulevaisuus

5.1 Toteuttamatta jäneet ominaisuudet

Robotti jäi kahdelta osin hieman keskeneräiseksi:

Ensinnäkin, peli ei osaa pelata ristinollaa täysin itsenäisesti, sillä BotGame:n hahmontunnuskoodi 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 sääntöjen mukaan, muttei erityisen älykkäästi.

Molemmat ongelmat olisi ollut tarkoitus ratkaista: Ristinollatekoälyn toteutus jonkinlaisella minimax-algoritmillä 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 nähdyn. Käyttäjän on sijoitettava robotti ennaltavalittuun asentoon peliruudukkoon nähdyn (pelilauden lävistäjän kautta kulkevalle suoralle). Mikäli robotti on hieman vinossa, se myös ajaa hieman sivuun ja pahimmassa tapauksessa piirtää ristejä väriin paikkoihin. Ongelmaa voi si hieman helpottaa laajentamalla nykyistä toiminnallisutta 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 pisteyiden 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 käytetty vakiot, raja-arvot, jne. on löydetty käsin kokeilemalla tietynlaisella kamerasuunnitelmalla. Esimerkiksi huomasin että tutkittavien kuvien resoluution vaihtaminen voi rikkota nykyisen toiminnallisuden.

6 Käyttöohjeet

6.1 Ohjelmistojen kääntäminen

6.1.1 Penbot

6.1.2 BotGame

6.2 Pelaaminen