Helsingin yliopisto
Tietojenkäsittelytieteen laitos

Cum laude laboratorio
Ohjelmistotuotantoprojekti Nobel

VAATIMUSDOKUMENTTI

Ilkka Forsblom, Risto-Matti Kääriäinen, Jussi Lahti,
Maarit Saaresto & Jarkko Salminen

versio 1.2

Ohjaaja:	**Teemu Kurppa**
Tilaaja:	**Tapio Elomaa**
Jätetty:	26.2.2001

Sisältö

1 Johdanto

1.1 Tuotteen tausta ja tarkoitus

Helsingin yliopiston Tietojenkäsittelytieteen laitokselle on joitakin vuosia sitten hankittu `Nomad Super Scout II`–robotti tutkimuskäyttöön. Sen sensoreita ovat pyöräodometria (kuljetun matkan ja rintamasuunnan arviointi pyörien pyörimisen perusteella), törmäyksen tunnistimet (6 kpl), ultraääniluotaimet (sonar, 16 kpl) sekä (ohjattava) kamera. Robotilla on kehittynyt graafinen ohjaus- ja simulointiympäristö, joka ei kuitenkaan tue robotin kaikkea sensori-informaatiota, mm. kamerakuvaa, eikä havaintokarttojen muodostamista sonar-sensoreiden antaman palautteen perusteella.

Toteutettavassa vaellusohjelmistossa robotin paikanmäärittämiseen käytetään aikaisemmin testattua menetelmää. Menetelmässä robotille annetaan kartta sen toimintaympäristöstä sekä joukko optisesti havaittavia maamerkkejä sijainteineen. Tämän lisäksi robotille annetaan maalipiste, johon sen tulisi suunnistaa. Odometrian avulla suoritetun vaelluksen tarkkuutta pyritään parantamaan määrittämällä sijainti uudelleen määräajoin maamerkeistä. Sijainti lasketaan triangulointimenetelmällä vähintään kolmesta yhtäaikaa havaitusta erillisestä maamerkistä. Maamerkit havaitaan ja niiden sijainnit robottiin nähden määritellään robotin videokameran avulla.

Toteutetun ohjelmiston suoritustarkkuutta tullaan käyttämään referenssituloksena uusia navigointimenetelmiä arvioitaessa.

1.2 Erikoissanasto ja käytetyt lyhenteet

este robotin liikkeen estävä objekti fyysisessä maailmassa.

fyysinen reitti robotin fyysisessä maailmassa kulkema reitti.

hätäpysäytys robotin pysähtyminen sonarien antaman datan perusteella.

kartta koordinaattijoukko, joka rajaa yhden tai useamman alueen robotille kiellettyksi alueeksi tai alueiksi. Ohjelma saa koordinaatit syötteenään.

kielletty alue alue, jolle robotin ei toivota kulkevan navigoinnin aikana. Kaikkien robotin ympäristössä olevien esteiden tulisi sijaita kiellettyjen alueiden sisäpuolella.

lasketut reitit reitit, jotka on laskettu maalipisteeseen sisäisellä kartalla.

liikkuminen robotin fyysinen liikkuminen.

maamerkki robotin ympäristöstään tunnistama hahmo, jonka koordinaatit suhteessa karttaan on sen tiedossa.

maalipiste kartan koordinaattipiste, johon ohjelmisto pyrkii robotin ohjaamaan.

etäisyys , joka vähintään jää navigoitaessa kiellettyjen alueiden ja robotin väliin.

Martti **m**obiili **a**utonominen **r**obo**tti**

monitorointitietokone robotin ulkopuolinen tietokone, johon robotti pitää yhteyttä joko modeemilla radiolinkin tai ethernet–kaapelin välityksellä.

navigointi liikkuminen, reitinlaskenta ja paikanmääritys yhdessä.

odometria kuljetun matkan ja rintamasuunnan arviointi pyörien pyörimisen perusteella

ohjelman keskeytys ohjelman keskeyttäminen SIGINT:llä (näppäimistöltä annettu keskeytys, CONTROL+C), joka ainoastaan pysäyttää robotin liikkumisen ja lopettaa ohjelman suorituksen.

ohjelman lopettaminen käyttäjän pyytämä (esim. johonkin näppäimeen sidottu) ohjelman hallittu lopettaminen, joka pysäyttää robotin. Pysäytyksen yhteydessä tehdään mm. tallennukset (kuvat, logit) ennen suorituksen lopettamista.

robotti tässä tapauksessa sonarein ja askelmoottorein varustettu tietokone.

segmentointi kuvan värialueiden erottaminen.

sensori havaintolaite, jonka avulla robotti tutkii ympäristöään. Projektin yhteydessä sensoreina käytetään odometriaa, ultraääniantureita ja videokameraa.

sisäinen kartta ohjelman sisäinen tietorakenne, joka sisältää (marginaaleilla laajennetut) kielletyt alueet sekä kaikki lasketut reitit maalipisteeseen

paikanmääritys sijainnin selvittäminen maamerkkien avulla.

pysäytysetäisyys robotin ylläpitämä minimietäisyys esteisiin. Robotin etäisyyttä esteisiin havainnoidaan sonareiden avulla. Esteeseen reagoidaan kun se havaitaan pysäytysetäisyyden sisäpuolella.

sonar ääntä heijastavien pintojen ja niiden etäisyyksien havainnointiin käytetty ultraääniluotain.

SVM **S**upport **V**ector **M**achine, tukivektorikone. Koneoppimismenetelmä, jota tuotettavassa ohjelmistossa käytetään maamerkkien hahmojen tunnistamiseen.

yllättävä este pysäytysetäisyyden sisällä oleva sonareiden havaitsema este.

1.3 Yleiskatsaus dokumenttiin

Luku 2: Kuvataan robotin ohjelmistolle tarjoama käyttöympäristö sekä määritellään käyttäjäkunta, jolle ohjelmisto on ensisijaisesti suunnattu.

Luku 3: Kuvataan ohjelmiston komponentteja vastaavien luokkien tietosisältöjen tarve.

Luku 4: Kuvataan ohjelmistolta vaadittuja toimintoja.

Luku 5: Kuvataan ohjelmiston ulkoiset liittymät eli käyttöliittymä, laitteistoliittymät ja ohjelmistoliittymät.

Luku 6: Kuvataan asiakkaan taholta esitetyt mm. ohjelmiston suorituskykyyn, käytettävyyteen ja ylläpidettävyyteen liittyvät vaatimukset.

Luku 7: Kuvataan joukko yleisiä tilanteita, joista robotin on ohjelmistoa suorittaessaan selviydyttävä.

Luku 8: Kuvataan robotin laitteiston ja käytettävien ohjelmistojen asettamia yleisiä rajoitteita ohjelmiston toteutuksen kannalta.

2 Yleiskuvaus

2.1 Yleinen toiminta

Ohjelmiston ajoa käynnistettäessä annetaan sille syötteenä kolme tiedostonimeä:

- ympäristön kiellettyjen alueiden kartan sisältävän tiedoston nimi,
- maamerkit sisältävän tiedoston nimi sekä
- ajokohtaiset asetukset ja maalipisteen tiedot sisältävän tiedoston nimi.

Ohjelman käynnistämisen jälkeen robotti etsii ensimmäisenä sijaintinsa kartalta havaitsemiensa maamerkkien avulla. Vaihtoehtoisesti aloituspiste voidaan kertoa robotille ajoasetusten kanssa samassa tiedostossa.

Maalipisteeseen navigoinnin lisäksi ohjelmisto pyrkii välttämään törmäykset fyysisiin esteisiin. Niitä tunnistetaan liikkumisen aikana ultraäänisensorien avulla. Onnistuessaan navigoinnissa robotin ei tulisi havaita kartan kielletyillä alueilla sijaitsevia esteitä lainkaan. Havaittuun esteeseen reagoidaan pysäyttämällä robotin liikkuminen. Mikäli este ei poistu ultraäänisensorien havaintopiiristä riittävän nopeasti, ohjelmisto joko pyrkii kiertämään esteen tai lopettaa ajonsa.

Aloituksessa saamiensa tietojen ja sensoridatan avulla ohjelmiston tulee ohjata robotti pyydetyn maalipisteen läheisyyteen. Robotin päätöksentekoketjua voi seurata ohjelmiston antamien tulosteiden pohjalta.

2.2 Pääasiallinen toimintaympäristö

Suunnistusohjelmistoa ajetaan Nomad Super Scout II robotissa. Robottia ohjaa 233 Mhz Intel Pentium MMX prosessorilla ja 256 MB muistilla varustettu tietokone. Kovalevyn kapasiteetti on 5,2 GB, minkä lisäksi robotti sisältää `Brooktree Bt848` kuvankaappauskortin ja Realtek 8029 piirillä varustetun verkkokortin. Kone voidaan liittää TKTL:n Linux–verkkoon joko suoraan ethernet–kaapelilla hubiin tai radiolinkin kautta radiomodeemin avulla. Käyttöjärjestelmänä robotin tietokoneessa on TKTL:n Red Hat 6.1 -pohjainen CSL2 (Linux version 2.2.17).

Robotti saa virtansa kahdesta akusta, joita se lataa automaattisesti olleessaan kytkettynä sähköverkkoon. Ajojen ulkopuolella robotti pidetään kiinni sähköverkossa.

2.3 Käyttäjäkunta

Toteutettava ohjelmisto on tarkoitettu ensisijaisesti Helsingin yliopiston tietojenkäsittelytieteen robotiikan tutkimusryhmän käyttöön.

2.4 Katsaus aikaisempiin ohjelmistoihin

Keväällä 2000 Tietojenkäsittelytieteen laitoksen Ohjelmistotuotantoprojekti –kurssin harjoitustyönä toteutettiin samalle robotille pääasiallisesti sonareita ja törmäyssensoreita hyödyntävä vaellusohjelmisto. Ohjelmiston tavoitteena oli muunmuassa ympäristön kartoittaminen vaeltamisen aikaisten sonar–havaintojen perusteella.

Toteutettu ohjelmisto pyrki myös tunnistamaan kamerakuvasta yksinkertaisia maamerkkejä ja määrittelemään niiden suunnan suhteessa itseensä.

3 Tietosisältö

3.1 Ohjelmiston syötteet

Käyttäjän on annettava ohjelmistolle syötteeksi kartta ympäröivän maailman esteistä ja maalipiste, johon robotin on navigoitava. Kartalle annetaan myös mittakaava, jotta ohjelma tietää esteiden todellisen koon. Maamerkit ja niiden sijainnit merkitään omaan syötetiedostoonsa. Näiden välttämättömien syötteiden oikeellisuus on käyttäjän vastuulla. Jos kartasta puuttuu esteitä, robotin navigointi saattaa estyä. Maamerkkien sijainnit pitää antaa tarkasti. Samanlainen maamerkki voi esiintyä robotin ympäristössä monta kertaa, mutta käyttäjän on aseteltava merkit eri paikkoihin, jottei robotti voi havaita kahta samanlaista merkkiä yhtäaikaa. Jos näin kuitenkin tapahtuu eikä muita korvaavia merkkejä ole näkyvissä, paikanmääritys epäonnistuu. Maamerkit on myös sijoiteltava järkevästi robotin kulkureittien ympärille, jotta robotin on mahdollista laskea sijaintinsa triangulointimenetelmällä. Vaikka paikanmääritys epäonnistuisikin jostakin syystä, navigointia voidaan silti jatkaa, mutta maalipisteen saavuttaminen ei välttämättä onnistu.

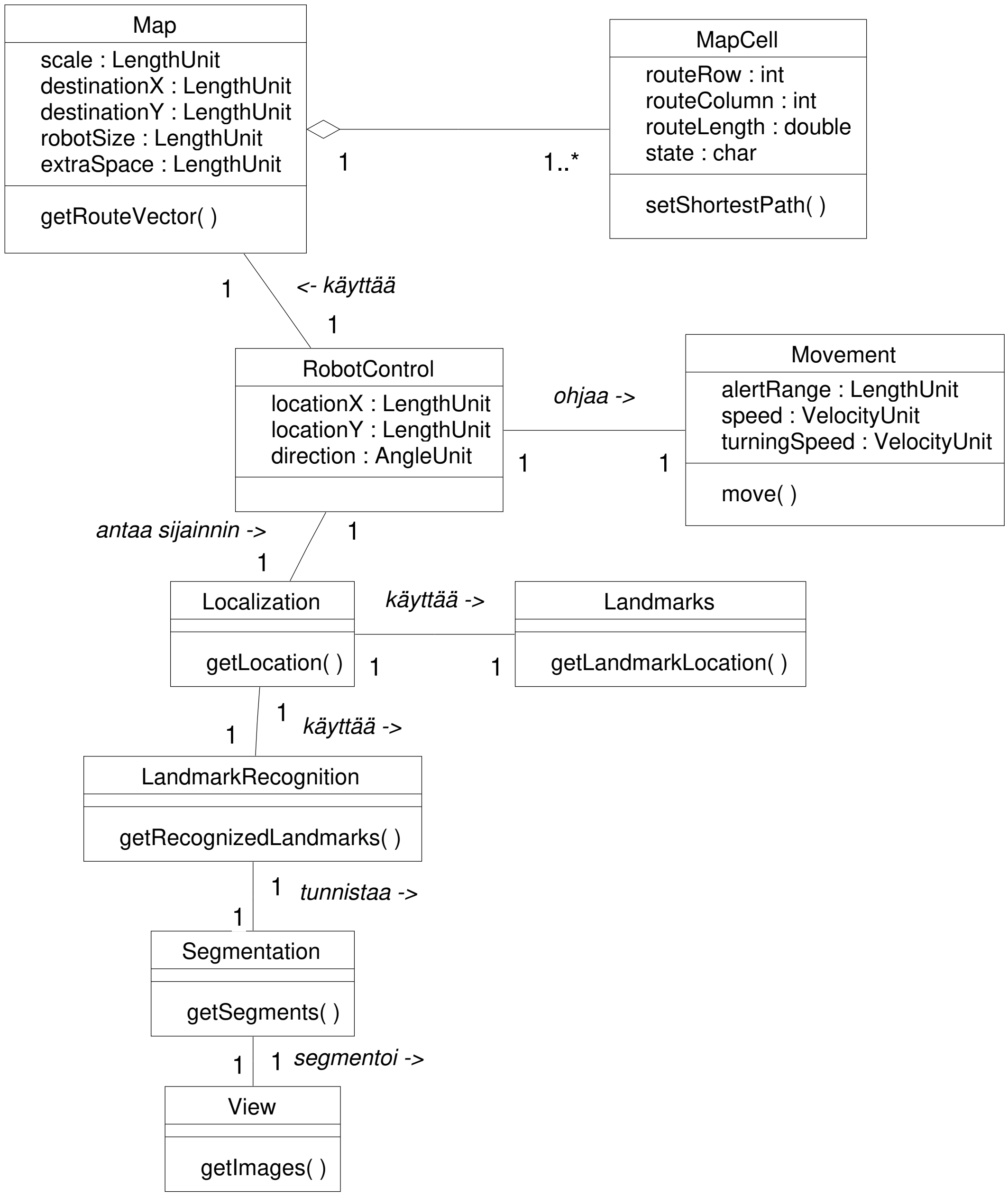
Käyttäjä voi halutessaan antaa ohjelmistolle parametreja, jotka ohjaavat laskentaa ja vaikuttavat järjestelmän tulosteisiin. Syötteenä voi antaa, kuinka usein robotti tarkistaa sijaintinsa maamerkkien avulla. Etenemis- ja kääntymisnopeudet ovat myös syötteitä. Käyttäjä voi määritellä kiellettyjen alueiden marginaalin ja sonareiden pysäytysetäisyyden suhteessa esteisiin. Robotti voi havaita sonareillaan yllättäviä esteitä, mutta se ei osaa kiertää niitä. Käyttäjä voi määritellä, kuinka kauan robotti odottaa reitin vapautumista. Jos este ei siirry, ohjelma lopetetaan ennenaikaisesti. Järjestelmä tallentaa kameran ottamat kuvat ja niistä havaitut maamerkit käyttäjän niin halutessa. Ohjelman voi lopettaa navigoinnin aikana.

3.2 Olioyhteyskaavio ja järjestelmän tietosisältö

Ohjausosajärjestelmä: `RobotControl`–luokka sisältää tiedon robotin sijainnista, kentät `locationX` ja `locationY`, ja suunnasta kartan suhteen, kenttä `direction`. `RobotControl` muodostaa oman osajärjestelmänsä, joka on ohjelmiston ydin. Se ohjaa koko järjestelmän toimintaa.

Reitinlaskentaosajärjestelmä: `Map`– ja `MapCell`–oliot sisältävät sisäisen kartan ja lyhimmät reitit kaikista kartan vapaista pisteistä maalipisteeseen. `Destination`–kentät kertovat maalin sijainnin. `Map`–luokan attribuutti `extraSpace` määrittelee marginaalin, jottei robotti liiku liian lähellä esteitä. Map-luokan pitää myös tietää robotin koko. `Scale` on kartan mittakaava. `Map`–olio antaa `RobotControl`–oliolle mahdollisimman pitkän suoran reitiltä, kun se saa parametrina robotin sijainnin. `MapCell`–oliot tallentavat reitit `routeRow` ja `routeColumn` kenttiin, ja `routeLength` on reitin pituus maaliin. `State` kuvaa olion ulkoasun ja merkityksen kartalla: `MapCell` voi olla este tai vapaata tilaa.

Liikkumisosajärjestelmä: `Movement`–luokka huolehtii fyysisestä liikkumisesta ja yllättävien esteiden havaitsemisesta. Tässä se käyttää hyväkseen sonareita, jotka



Kuva 3.1: Olioyhteyskaavio järjestelmän tietosisällöstä

varoittavat hälytysetäisyyden (`alertRange`) sisällä olevista esteistä. Nopeusattribuutit ohjaavat etenemistä ja kääntymistä.

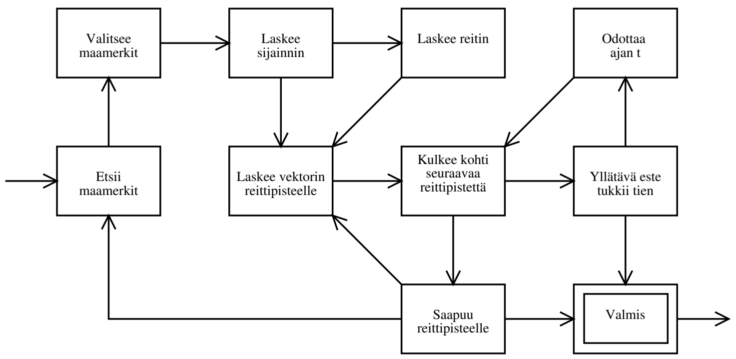
Paikanmääritysosajärjestelmä: `Location` laskee robotin tarkan sijainnin ja suunnan maamerkkien perusteella. Se käyttää hyväkseen `Landmark`–olioiden tietoja maamerkkien sijainneista (`locationX`, `locationY`). `LandmarkRecognition` ottaa kuvia robotin ympäristöstä ja tunnistaa niissä esiintyvät maamerkit. Luokka käyttää hyväksi tukivektorikonetta, joka käsittää tiedon maamerkkien hahmoista. `Image`–oliot sisältävät tiedot otetuista kuvista: nimi, kulma, jossa kuva otettiin, ja varsinainen kuvadata.

3.3 Ohjelmiston tulosteet

Ohjelmisto tulostaa järjestelmän laskentaa ja tilaa kuvaavia tietoja. Kartta ja lasketut reitit tulostetaan käyttäjälle merkkigrafiikkana havainnollistamaan robotin navigointia. Lasketut osareitit annetaan myös numeerisessa muodossa koordinaatteina. Sonarsensorin havaitsemat yllättävät esteet raportoidaan. Kameran ottamat kuvat ja hahmosegmentit tallennetaan robotin kovalevylle. Havainnot maamerkeistä sekä paikanmäärityksen perusteella lasketut odometrian virheet tulostetaan. Samoin tiedot poikkeustilanteista, robotin sijainnista ja suunnasta ilmoitetaan käyttäjälle. Järjestelmä tallentaa kaikki teknisetkin tulosteet omaan log-tiedostoonsa. Sen sijaan käyttäjä voi määritellä kuinka paljon tietoja ohjelman käyttöliittymä välittää. Käyttöliittymän tulosteetkin voi uudelleenohjata erilliseen tiedostoon. Tulosteiden kieli on englanti.

3.4 Kapasiteetti- ja saantiaikavaatimukset

Robotti ottaa kuvia ympäristöstään etsiessään maamerkkejä. Näiden kuvien tallentamiseen tarvittava kovalevytila on vähäinen verrattuna robotin levyn kapasiteettiin. Keskusmuistia robotissa on 256 megatavua, joka on riittävä kartan ja kuvien käsittelyyn. Robotin radiolinkin tiedonsiirtonopeus on liian heikko kuvien siirtämiseen ajon aikana, minkä takia kuvat tallennetaan levylle. Navigointitehtävän suorittamiselle ei ole asetettu erityisiä aikavaatimuksia, mutta tavoitteena on saada robotti toimimaan mahdollisimman ripeästi. Reitin laskeminen ja paikanmääritys maamerkkien avulla ovat laskennallisesti raskaimmat vaiheet. Pelkästään kameran liikuttelussa kuluu aikaa. Kuvankäsittelyssä käytetään valmiita kirjastoja, joten niiden suorituskykyyn ei voi vaikuttaa. Reitin laskeminen Dijkstra–algoritmilla on kuitenkin riittävän nopeaa (alle 10 sekuntia), ja se pitää suorittaa vain kerran.



Kuva 4.1: Tilasiirtymäkaavio

4 Toimintokuvaus

Robotin korkean tason toimintoa kutsutaan navigoinniksi. Se voidaan jakaa kolmeen osatoimintoon, jotka ovat reitinlaskenta, paikanmääritys ja liikkuminen. Jokainen osatoiminto voidaan edelleen osittaa hienompiin toimintoihin. Järjestelmän yleisarkkitehtuurissa näitä kolmea osatoimintoa vastaa oma osajärjestelmänsä. Jokainen osajärjestelmä on myös ositettu, joista tarkentuu järjestelmän oliomalli. Lisäksi järjestelmä sisältää navigoinnin ulkopuolisia toimintoja, joista ensimmäinen on järjestelmän suorituksen keskeytys, joka pysäyttää navigoinnin so. myös liikkumisen. Toinen navigointiin liittymätön toiminto on tulostus, joka riippuen käyttäjän alussa antamista syötteistä antaa ajonaikaista informaatiota järjestelmän toiminnasta.

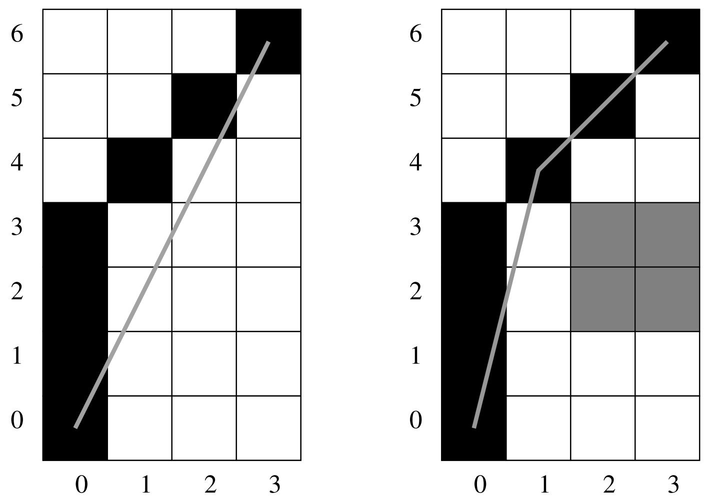
Järjestelmän yleistä toimintaa voidaan kuvata tilasiirtymäkaaviolla, (kuva 4.1), jossa jokaisesta tilasta päästään toiseen tietyn toiminnon kautta. Tilasiirtymäkaavio ei ota kantaa osajärjestelmiin eikä varsinkaan oliomalliin.

4.1 Reitinlaskenta

Reitinlaskenta sisältää syötteenä annetun kiellettyjen alueiden kartan hajoittamisen soluihin Dijkstra-reitinlaskualgoritmia[1] varten. Samalla kartan kiellettyjä alueita laajennetaan syötteenä annetun marginaalin verran. Dijkstran algoritmilla löydetään lyhin reitti kaikista mahdollisista soluista syötteenä annettuun maalipisteen soluun. Käytännössä jokaisessa solussa on viittaus siihen viereiseen soluun, jota kautta lyhin reitti maaliin muodostuu.

Dijkstran algoritmi ei kuitenkaan löydä optimaalisen lyhyitä reittejä, koska se mahdollistaa liikkumisen ainoastaan kahdeksaan ilmansuuntaan eli pysty-, vaaka- ja diagonaalisuuntiin. Seuraava esimerkki näyttää, miten Dijkstra–algoritmin löytämiä reittejä voidaan edelleen optimoida:

Oletetaan, että Dijkstran algoritmi on löytänyt reitin, (vasen kuva 4.2),



Kuva 4.2: **Vasemmalla:** Dijkstran reititys ilman kiellettyä aluetta. **Oikealla:** Dijkstran reititys kielletyllä (harmaalla) alueella.

> jossa liikutaan solusta (0,0) 3 askelta pohjoiseen (N) ja sitten 3 koilliseen (NE) soluun (3,6). Solusta (0,0) on olemassa lyhyempi (lyhin) reitti soluun (6,3), joka kulkee suoraan (linnuntietä). Optimoinnissa täytyy kuitenkin tarkistaa, että suora reitti ei ylitä kartalla yhtään sellaista solua, joka on ns. kielletty alue, (oikea kuva 4.2). Jos optimaalisin reitti ei ole mahdollinen so. se ylittää kiellettyjä alueita, voidaan etsiä yhä vähemmän optimaalista reittiä (esim. reitti solun (4,1) kautta). Jos optimointi ei onnistu, täytyy tyytyä alkuperäiseen Dijkstran löytämään reittiin, joka ei ylitä kiellettyjä alueita.

Reitinlaskussa syntyy virhetilanne, jos robotti sijaitsee kielletyllä alueella. Tällöin, joko kiellettyjen alueiden kartta on virheellinen, tai paikanmääritys on epäonnistunut. Tilanteessa voidaan suorittaa ns. random walk, joka liikuttaa robottia esteettömään suuntaan. Random walk suoritetaan n kertaa uudestaan (n on määritelty ajotiedostossa), jos robotti sijaitsee edelleen kielletyllä alueella. Jos kaikkien yritystenkin jälkeen robotti sijaitsee kielletyllä alueella, ohjelman suoritus lopetetaan.

Toisen tyyppinen virhetilanne syntyy, kun robotin sijaintipisteestä ei löydy reittiä maaliin. Tällöin ohjelman suoritus vain lopetetaan.

4.2 Paikanmääritys

Paikanmääritys perustuu maamerkkien etsimiseen robotin kameralla ja niistä sijainnin laskemiseen. Sijainti lasketaan ns. triangulointimenetelmällä[2], jossa kolmen robotin ympäristössä havaitun maamerkin (ja niiden kulmien suhteessa robottiin) avulla voidaan robotin sijainti määritellä. Maamerkkien havaitseminen ja tunnistaminen kamerakuvasta vaati kuvien värialueiden segmentointia ja niiden hahmon tunnistamista. Maamerkkien hahmot on opetettu järjestelmälle valmiiksi ja niiden sijainnit kartalla tunnetaan. Värialueiden segmentointiin ja niiden hahmojen tunnistamiseen käytetään valmiita kirjastoja.

Paikanmäärityksessä virhetilanne syntyy, jos maamerkkejä ei löydy ympäristöstä kahta enempää. Tilanteessa voidaan suorittaa ns. random walk, joka liikuttaa robottia esteettömään suuntaan. Random walk suoritetaan n kertaa uudestaan (n on määritelty ajotiedostossa), jos robotti ei edelleenkään löydä kahta maamerkkiä enempää. Jos kaikkien yritystenkin jälkeen robotti on kykenemätön löytämään maamerkkejä kahta enempää, ohjelman suoritus lopetetaan.

Toinen virhetilanne syntyy, jos robotti löytää kolme maamerkkiä (tai enemmän), mutta ei sijaitse minkään kolmen maamerkin muodostaman kolmion sisäpuolella (trianguloinnilla ei saada silloin luotettavaa tulosta). Tällöin robottia liikutetaan siten, että se lopulta sijaitsee kolmion sisäpuolella.

4.3 Liikkuminen

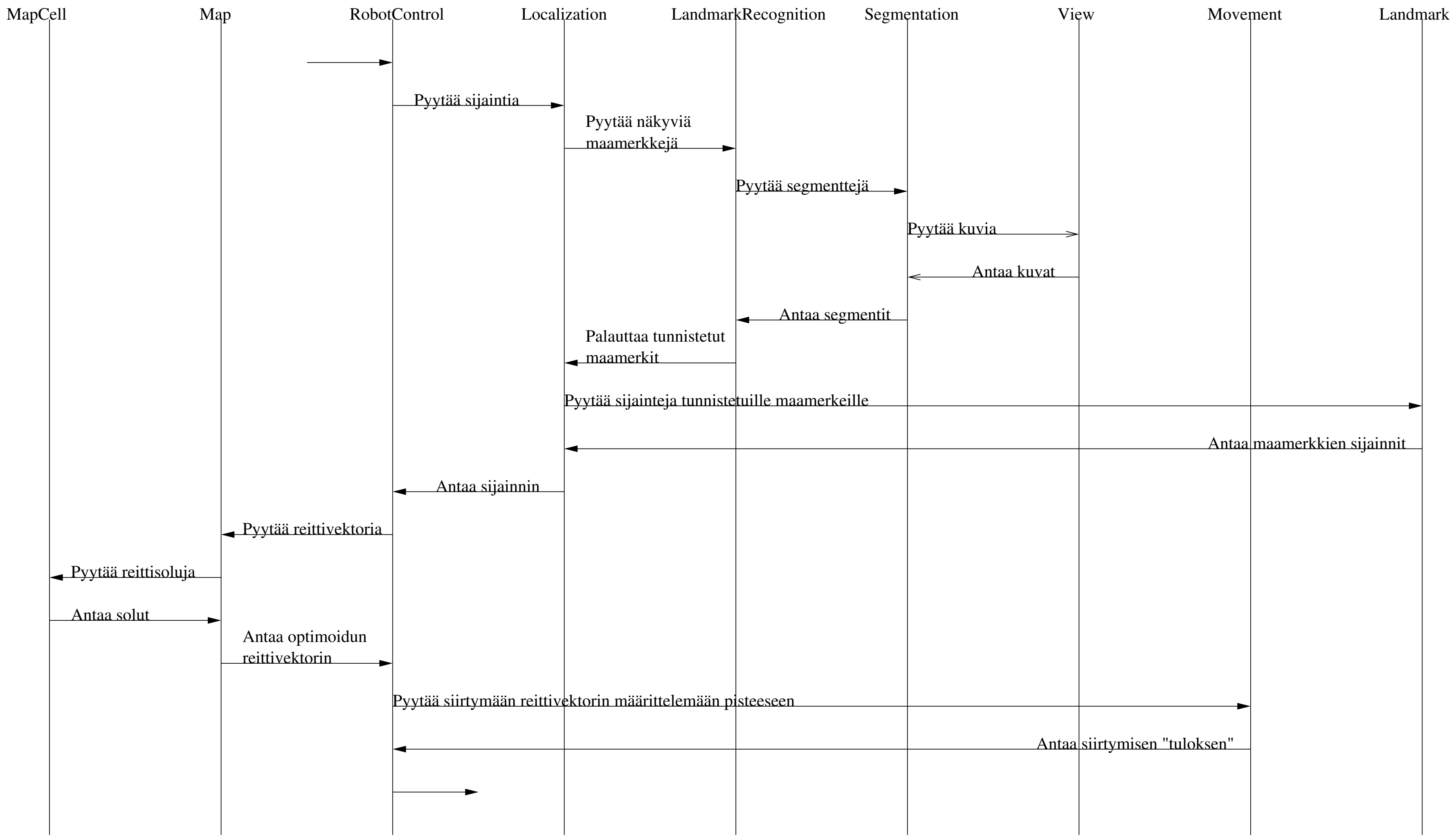
Liikkuminen on varsin yksinkertainen toiminto, joka osaa annetun suuntavektorin mukaisesti kääntää robottia ja kulkea eteenpäin. Liikkumiseen liittyy myös yllättävän esteen havaitsemisesta (so. sonarien havainnosta) johtuva hätäpysäytys. Pysäytystä seuraa joko ohjelman suorituksen keskeytyminen, odotus tai ns. random walk, joka liikuttaa robottia vapaaseen suuntaan. Ideaalisessa tilanteessa sonarit eivät havaitse mitään, koska reitin tulisi kulkea kiellettyjen alueiden ulkopuolla, eikä kiellettyjen alueiden ulkopuolella pitäisi olla esteitä. Käytännössä paikanmääritys tai liikkuminen voi olla epätarkkaa, jolloin robotti saattaa ajautua kielletyille alueille. Myös kiellettyjen alueiden kartta saattaa olla puutteellinen. Myös liikkuvat esteet so. ihmiset (ja muut kotieläimet) saattavat tuottaa hankaluuksia.

4.4 Skenaariokaaviot

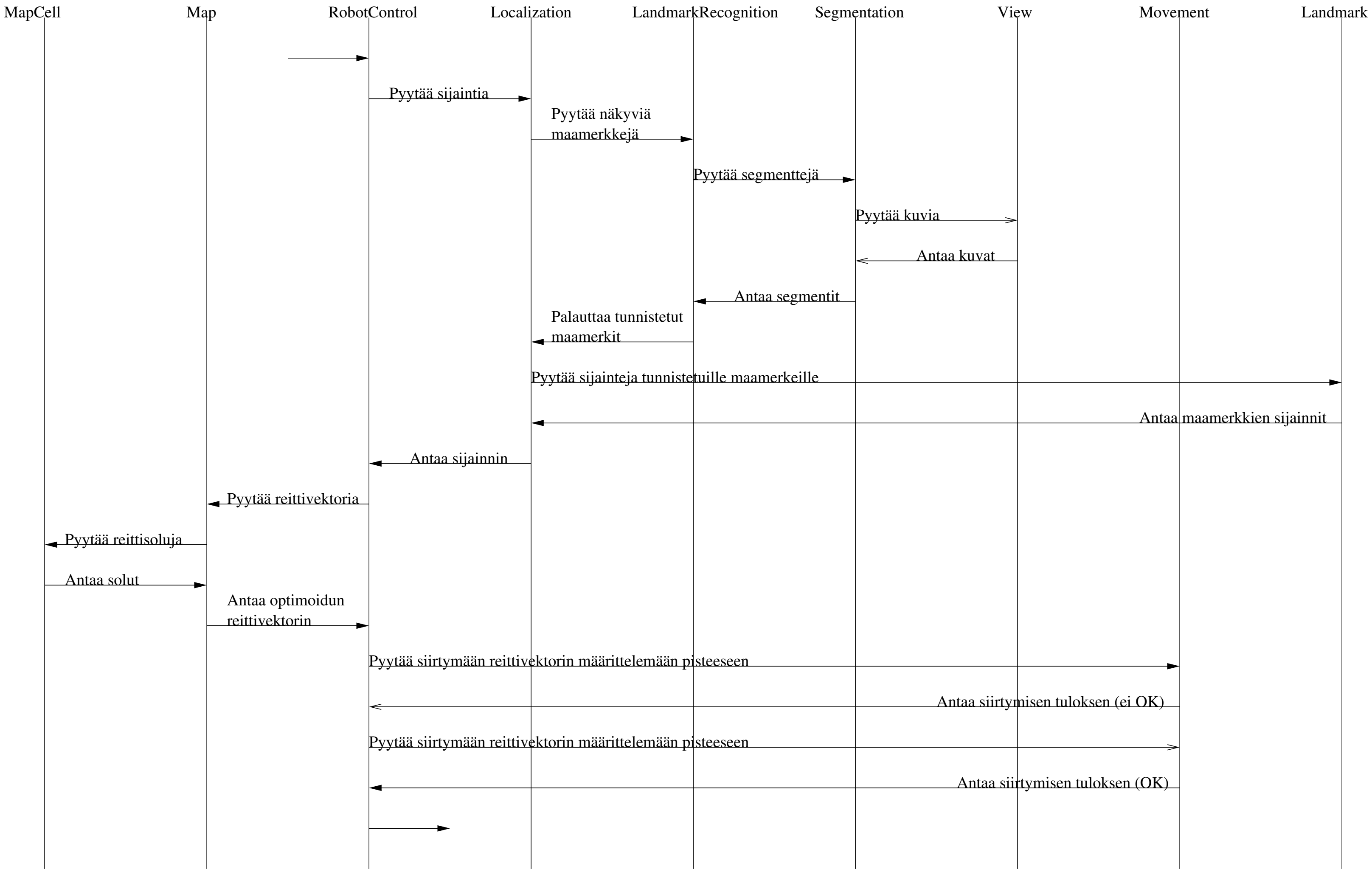
Järjestelmän toimintoa on kuvattu myös kahdella skenaariokaaviolla. Niissä on kuvattu toimintoja olioiden kautta so. miten oliot toimivat keskenään ja kutsuvat toisiaan. Kummassakin skenaariossa käydään läpi yksi mahdollinen lineaarinen ketju toimintoja, ensimmäisessä, (kuva 4.3), suoritetaan onnistunut navigointi ilman yllättävien esteiden havaitsemista ja toisessa, (kuva 4.4), havaitaan yllättävä este, mutta siitä toivutaan eli odotetaan kunnes este katoaa ja jatketaan matkaa.

4.5 Yleisarkkitehtuuri

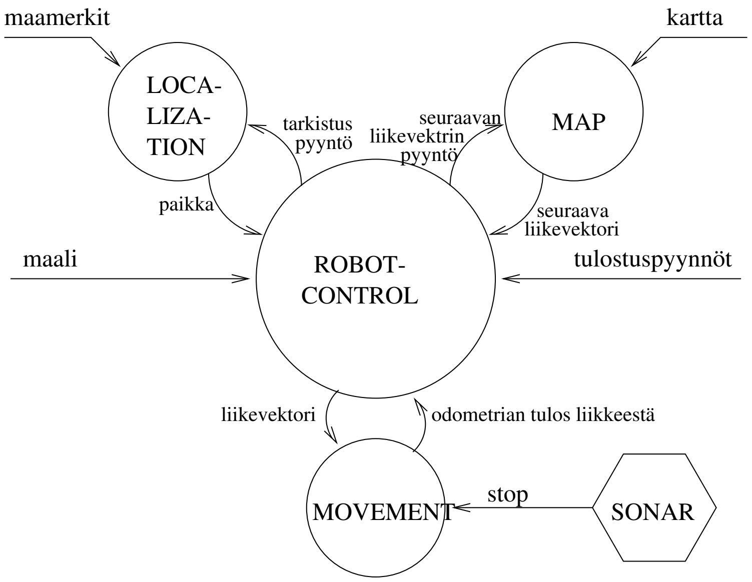
Järjestelmän yleisarkkitehtuurissa, (kuva 4.5), on järjestelmä ositettu neljään osajärjestelmään, joista kolme vastaa edelläkuvattuja kolmea perustoimintoa, reitinlaskentaa, paikanmääritystä ja liikkumista. Lisäksi on järjestelmän ydin, joka hallitsee kolmea muuta osajärjestelmää. Kaikki osajärjestelmien välinen informaatio- ja ohjausvirta kulkee ytimen kautta. Osajärjestelmien tarkempi ositus näkyy ja havainnollistuu oliomallista ja skenaariokaavioista.



Kuva 4.3: Olio-toiminto -kaavio, ei yllättäviä esteitä



Kuva 4.4: Olio-toiminto -kaavio, yllättävä este, josta toivutaan



Kuva 4.5: Yleisarkkitehtuurin korkean tason ositus

5 Järjestelmän ulkoiset liittymät

5.1 Käyttöliittymä

Ohjelmiston käyttöliittymä on tekstipohjainen. Ohjelma käynnistetään komentoriviltä ja sille annetaan parametreinä kolme datatiedostoa:

1. Karttatiedosto, joka koordinaattijoukoin ilmaisee kartan alueen kielletyt alueet.
2. Maamerkkitiedosto, joka sisältää käytettävien maamerkkien koordinaatit kartalla ja viitteet niitä vastaaviin tukivektorikoneiden tiedostoihin.
3. Ajokohtainen asetustiedosto. Tämä sisältää navigoinnin maalipisteen, johon robotin on kuljettava, sekä kaikki käyttäjän asetukset:
   - tulostuksen asetukset,
   - pysäytysetäisyys senteissä,
   - marginaali senteissä,
   - reagointipolitiikka suhteessa yllättäviin esteisiin,
   - segmentoinnin eli maamerkkien värirajat,
   - kartan mittakaava senteissä

Ohjelmiston ajonaikainen käyttöliittymä rajoittuu ohjelmiston tulosteisiin ja käyttäjän mahdolliseen lopetus- tai keskeytyskäskyyn. Robotti toimii ohjelmiston turvin autonomisesti käynnistyksen jälkeen, kunnes suoritus päättyy robotin saavuttua maalipisteen.

Ajonaikaiset tulosteet on tarkoitettu ohjelmiston logiikan seuraamiseen ja mahdollisten toimintavirheiden tarkkailuun. Ohjelmisto tulostaa käyttäjän asetusten mukaisesti tietoa maamerkkien tunnistamisyrityksistä ja niiden tuloksista, lasketusta reitistä ja positosta, havaituista odometriavirheistä ja niiden korjausmanöövereistä sekä muista merkittävistä valintatilanteista. Tulostuksille annetaan ajotiedostossa taso, joka määrää kuinka paljon tulostetaan. Lisäksi määritellään mikä tai mitkä osajärjestelmät tulostavat.

Ajonaikaiset tulosteet ohjataan erilliseen lokitiedostoon ja asetusten mukaan osin reaaliaikaisesti radiolinkin kautta monitorointikoneen näytölle. Kamerakuvaa tai segmentointituloksia voidaan haluttaessa tallentaa robotin kiintolevylle. Robotin monitorointikoneeseen yhdistävän radiolinkin rajallisen kapasiteetin takia kamerakuvan reaaliaikainen päivittäminen ei kuitenkaan ole mahdollista.

5.2 Laitteistoliittymät

5.2.1 Pyörät ja odometria

Robottia liikutetaan antamalla käskyjä pyörien askelmoottoreille. Paikallaan kääntyminen toteutetaan pyörittämällä pyöriä eri suuntiin.

Pyörien odometriasensorit antavat dataa pyörien kulkemasta matkasta. Tästä voidaan laskea robotin sijainnin muutos kartalla. Odometrian lukemia tarkistetaan paikanmäärityksen avulla.

Pyörien askelmoottorit toimivat epäsynkronisesti ohjelmiston kanssa, toisin sanoen askelmoottorien liikuttaessa robottia suoritetaan ohjelmistoa rinnakkaisesti. Tällöin voidaan lukea esimerkiksi sonarien dataa liikkumisen aikana.

5.2.2 Sonarit

Robotissa on 16 sonaria, joilla voidaan havaita lähistöllä olevat esteet. Sonarien laukaisutiheys ja laukaisujärjestys voidaan asettaa halutuksi. Sonarit toimivat itsenäisesti ohjelmistosta riippumatta päivittäen niinsanotun `state`–vektoriin kunkin sonarin mahdollisesti havaitseman pinnan etäisyyden, tai maksimietäisyyden jos kyseinen sonar ei havaitse pintoja.

Ohjelmisto käyttää sonareita yllättävien esteiden havaitsemiseen.

5.2.3 Törmäyssensorit

Robotissa on törmäyssensorit, joilla havaitaan robotin fyysinen kosketus esineisiin. Tuotettava ohjelmisto ei käytä törmäyssensoreita.

5.2.4 Kamera ja kuvankaappauskortti

Robottiin on asennettu videokamera sekä kuvankaappauskortti, joilla kamerakuva saadaan ohjelmiston käyttöön. Navigointi perustuu kameran näkemien maamerkkien tunnistamiseen ja sen avulla tapahtuvaan paikanmääritykseen.

5.2.5 Videokameran kuvauskulman muuttaminen

Kamera on asennettu pan-tilt –yksikköön, jolla kamera saadaan kääntymään pysty- ja vaakasuunnassa. Paikanmäärityksessä robotti pysyy paikallaan ja kameraa käännetään ympäri, jotta saadaan täyden panoraaman muodostava kuvasarja. Pystyliikettä ei käytetä.

5.3 Ohjelmistoliittymät

Järjestelmä käyttää hyväkseen kolmea koodikirjastoa robotin liikuttamiseen, kuvien segmentointiin ja hahmojen (maamerkkien) tunnistamiseen. Kirjastot sijaitsevat levypalvelimella hakemistossa: `/home/group/robotics/libs`. Robotin liikutuskirjasto on alihakemistossa `nomad`[3] , segmentointi `helvision` –hakemistossa (CMVisionin[4] paranneltu versio) ja hahmojen tunnistus `SvmFu_2.004` –hakemistossa[5]. Myös kameran kääntelylle on oma kirjastonsa, joka sijaitsee Martissa hakemistossa `/usr/local/robot-devices/dp_pantilt`. Kuvankaappaus onnistuu rajapinnan `/usr/include/linux/videodev.h`[6] kautta.

5.4 Tietoliikenneliittymät

Robotissa on radioyhteys monitorikoneeseen, jolta ohjelmisto käynnistetään. Ajon aikana radiolinkin kautta siirretään ajonaikaisia tulosteita ja tarkkaillaan mahdollisia käyttäjän komentoja.

6 Muut ominaisuudet

6.1 Suorituskyky

Suorituskyvylle ei ole asetettu tarkkoja vaatimuksia. Kuitenkin projektiryhmän tavoitteena on luoda ohjelmisto, joka on tehokas siinä mielessä, että valitut algoritmit ja toteutustavat eivät tarpeettomasti eivätkä merkittävästi hidasta robotin toimintaa. Ohjelmiston tulisi siten olla käytetyssä laitteistossa niin nopea, että suuremman rajoitteen robotin toiminnan nopeudelle antaa robotin liikkumisen ja kameran hitaus, ei algoritmien laskenta. Asiakas ei antanut tarkkoja vaatimuksia suorituskyvylle.

6.2 Käytettävyys, virheistä toipuminen, turvallisuus

Ohjelmistoa tullaan tulevaisuudessa käyttämään vertailupohjana muille samankaltaisille järjestelmille. Sen takia on erityisen tärkeää, että ohjelmisto toimii luotettavasti ja oikein, olettaen että ohjelman saamat syötteet ovat tarpeeksi hyvät. Annettaessa todellisuutta hyvin kuvaava kartta ja tarkat (hyvin sijoiteltujen) maamerkkien sijainnit syötteenä ohjelmistolle, robotin tulee pystyä suunnistamaan annettuun maalipisteeseen tarkasti ja luotettavasti.

Ohjelmistolta ei edellytetä varsinaista virheistä toipumista. Riittää, että virhetilanteen havaitessaan tai sellaiseen ajautuessaan ohjelmisto pystyy lopettamaan toiminnan luotettavasti. Ohjelmistolta edellytetään, että se pystyy havaitsemaan yllättävät esteet, eli esteet, joita ei ole merkitty ohjelmiston syötteenä saamalle kartalle, ja välttämään törmäykset pysäyttämällä robotin.

6.3 Ylläpidettävyys

Ohjelmistolta edellytetään selkeää ja toimivaa modulaarisuutta. Ohjelmiston osien tulee olla helposti korvattavissa muilla saman rajapinnan toteuttavilla osilla. Rajapintojen tulee olla selkeitä ja hyvin dokumentoituja. Tämä on erityisen tärkeää, koska on mahdollista, että ohjelmistoa tullaan jatkokehittämään muiden toimesta.

7 Testaus

Tuotettava ohjelmisto on monella tapaa kokeellinen. Aihealueena robotiikka ja siihen liittyvät ongelmat eivät ole projektiryhmän jäsenille kovinkaan tuttuja. Nämä ongelmat vaativat, että testausta on suoritettava jo erittäin varhain ja erittäin paljon.

Jo suunnitteluvaiheessa vaaditaan testausta. Suunnittelupäätösten toimivuutta tulee testata paperilla, jotta voidaan varmistua siitä, että ohjelmisto voisi edes teoriassa toimia robotissa. Jos suunnittelupäätöksen toimivuus riippuu kirjastosta tai muusta toiminnallisuudesta, jonka toimivuudesta ei ole varmuutta, on mahdollisuuksien mukaan rakennettava nopeasti testiohjelma, jolla suunnittelupäätöksen toimivuutta voidaan testata käytännössä.

Ohjelmistoa toteutettaessa komponenttien toimivuutta tulisi testata mahdollisimman aikaisessa vaiheessa. Erityisesti tulee edellyttää, että komponenttien yhteensopivuutta ja ohjelman toimimista kokonaisuutena tutkitaan testaamalla niin aikaisessa vaiheessa kuin mahdollista, jotta mahdollisesti ilmitulevien ongelmien ratkaisuun voidaan varata riittävästi aikaa.

Valmiin ohjelmiston testauksessa voidaan käyttää esimerkiksi seuraavanlaisia vaikeutuvia testitapauksia:

- Robotti sijoitetaan johonkin robottilaboratoriossa, ja ohjelmiston tehtävä on kertoa robotin sijainti maamerkkien avulla. Erilaisia tämäntapaisia tapauksia voidaan käyttää paikannusjärjestelmän toimivuuden testaamiseen, vaikka ohjelmisto muuten ei olisi valmis.
- Robotti sijoitetaan johonkin robottilaboratoriossa, ja sille annetaan maalipiste, joka sijaitsee robottilaboratoriossa. Robotin tulee maamerkkien avulla paikantaa itsensä, laskea reitti maalipisteeseen, kulkea reitti törmäämättä esteisiin, ja lopuksi tarkistaa olevansa oikeassa paikassa. Jos paikanmääritys maamerkkien avulla ei toimi, voidaan aloitussijainti antaa robotille, ja näin testata reitinlaskennan ja pelkän odometrian avulla suunnistamisen toimivuutta.
- Robotti sijoitetaan johonkin robottilaboratoriossa, ja sille annetaan maalipiste, joka sijaitsee laboratorion ulkopuolella olevalla käytävällä. Kaikki maamerkit ovat erilaisia. Robotin tulee toimia kuten edellä.
- Kuten edellä, mutta käytävän puolella ja robottilaboratoriossa käytetään samanlaisia maamerkkejä (kuitenkin niin, että paikanmääritys pystyy selvittämään robotin sijainnin yksiselitteisesti).

8 Rajoitteet suunnittelulle ja toteutukselle

8.1 Laitteistorajoitteet

Ohjelmiston tulee toimia laitoksen Nomad Super Scout II -robotissa. Robotin laitteiston toiminnan nopeus ja luotettavuus, tai niiden puute, sekä saatavilla olevan prosessointitehon määrä täytyy ottaa huomioon ohjelmistoa suunniteltaessa ja toteutettaessa. Lisäksi käyttöliittymän suunnittelussa ja toteutuksessa on otettava huomioon se, että käyttöliittymä siirretään hitaan radiolinkin yli.

8.2 Ohjelmistorajoitteet

Tuotettavan ohjelmiston tulee toimia laitoksen Nomad Super Scout II -robottiin asennetussa Linux-käyttöjärjestelmässä. Tarjolla on Linuxin vakiokirjastojen lisäksi robotin kanssa kommunikointiin ja robotin ohjaukseen käytettävä kirjasto, sekä erinäisiä kirjastoja kameran ja kuvankaappauskortin käyttöön, kuvien segmentointiin ja hahmontunnistukseen. Näiden kirjastojen toiminta ja niissä mahdollisesti esiintyvät puutteet on huomioitava ohjelmistoa suunniteltaessa ja toteutettaessa. Kirjastojen rajapinnat ja toteutus ovat C-kielisiä (jotkin mahdollisesti C++-kielisiä), mikä rajoittaa tuotettavan ohjelmiston toteutuskielen valintaa (joko C tai C++), mikä puolestaan vaikuttaa suunnitteluun ja toteutukseen.

Kirjallisuutta

[1] M. Weiss. *Data Structures and Algorithm Analysis*. The Benjamin/Cummings Publishing Company, USA, 1995.

[2] Charles Cohen and Frank V. Koss. A comprehensive study of three object triangulation. In *Proc. of the 1993 SPIE Conf. on Mobile Robots*, pages 95–106, November 1993.

[3] Nomadic Technologies. *Nomad Scout Language Reference Manual*. USA, 1999.

[4] The CORAL Group's Color Machine Vision Project at Carnegie Mellon School of Computer Science. Cmvision 1.1, 2000. http://www.cs.cmu.edu/%7Ejbruce/cmvision/.

[5] Ryan Rifkin and MIT. Svmfu version 2.004, 2000. http://five-percent-nation.mit.edu/SvmFu/.

[6] Video4linux. http://roadrunner.swansea.linux.org.uk/v4l.shtml.