Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto

Diplomityö

Kaukokäyttöliittymä liikkuvalle robotille

Diplomityön aihe on hyväksytty Tietotekniikan osastoneuvoston kokouksessa 18.5.2005. Työn tarkastajat ovat professori Heikki Kälviäinen ja dosentti, TkT Ville Kyrki. Työn ohjaaja on dosentti, TkT Ville Kyrki.

Lappeenrannassa 23. marraskuuta 2006

Jukka Turunen
Kierniemenraitti 6 A 10
53850 Lappeenranta
Puh. 040 7419742

Tiivistelmä

Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto
Turunen, Jukka
Kaukokäyttöliittymä liikkuvalle robotille
Diplomityö
2006
49 sivua, 8 kuvaa ja 1 taulukko

Tarkastajat: professori Heikki Kälviäinen ja dosentti, TkT Ville Kyrki
Hakusanat: robotiikka, kaukokäyttö, käyttöliittymä, etäkäyttö, liikkuvat robotit, langattomat lähiverkot, olio-ohjelmointi, wlan, gui
Keywords: mobile robotics, teleoperation, user interfaces, remote control, wireless local area networks, object-oriented programming, wlan, gui

Lappeenrannan teknillinen yliopiston Tietotekniikan osaston Tietojenkäsittelytieteen laitoksen tutkimuskäytössä olevaan liikkuvaan robottiin toteutettiin tässä työssä graafinen kaukokäyttöliittymä. Työlle on motivaationa laajennettavuus, jota olemassaoleva suljetun lähdekoodin käyttöliittymä ei pysty tarjoamaan.

Työssä olennaisin on olio-ohjelmointitekniikalla toteutettu robotin datamallin, ja sen graafisen esityksen arkkitehtuurillinen erottaminen. Lisäksi tarkastellaan lyhyesti liikkuvien robottien kaukokäyttöliittymien teoriaa, ja WLAN-tekniikan soveltuvuutta robotin ja käyttöliittymän välisen yhteyden toteuttamiseen.

Abstract

Lappeenranta University of Technology
Department of Information Technology
Turunen, Jukka
Teleoperation user interface for a mobile robot
Master's thesis
2006
49 pages, 8 figures and 1 table

Supervisors: professor Heikki Kälviäinen and docent, Dr. Tech. Ville Kyrki
Keywords: mobile robotics, teleoperation, user interfaces, remote control, wireless local area networks, object-oriented programming, wlan, gui

The goal for this master's thesis was to design and implement a graphical teleoperation interface for a mobile robot. The Department of Information Technology in Lappeenranta University of Technology has a mobile robot in use for research. Current interface is a closed-source application and thus non-expandable. This is the motivation for customisable interface application.

The architectural separation of robot data and its graphical presentation using object-oriented approach is the most important part of this diploma thesis. Research concerning teleoperation interfaces in mobile robot applications is presented. The applicability of WLAN technology for connecting the interface and the robot is also discussed.

Sisältö

1 Johdanto **5**

2 Robotiikkaa **8**

2.1 Kaukokäytön periaatteet 8

2.2 Kaukokäytön sovellukset 9

2.3 Kaukokäyttöliittymät 9

2.4 Odometrinen paikannus 10

2.5 Kaikuluotaus 11

2.6 Reitinsuunnittelu 11

3 Langaton lähiverkko **13**

3.1 Toiminta 13

3.2 WLAN kaukokäytössä 14

4 Graafiset käyttöliittymät **15**

5 Vaatimukset **17**

5.1 Linux-ympäristö 17

5.2 Arkkitehtuuri 18

5.3 Dokumentaatio 18

5.4 MobileEyes-sovelluksen toiminnallisuus 18

5.5 Graafinen asetusten hallinta 19

5.6 WLAN-yhteyden tila 20

5.7 Puhesyntetisaattorin ohjaus 20

6 Arkkitehtuuri **21**

6.1 ARIA-kirjasto . . . 21
6.2 Malli-näkymä . . . 21
6.2.1 Rajapinnat . . . 22
6.2.2 Malli . . . 22
6.2.3 Asetusten hallinta . . . 24
6.2.4 Näkymä . . . 24
6.3 Palvelin . . . 28
6.3.1 Videopalvelin . . . 29
6.3.2 Ohjauspalvelin . . . 29

7 Toteutus **30**

7.1 Rajapinnat . . . 30
7.2 Mutex-luokkamalli . . . 30
7.3 Malli . . . 31
7.3.1 Robotilta saatava tieto . . . 31
7.3.2 Komennot . . . 31
7.4 Näkymä . . . 32
7.4.1 Karttanäyttö . . . 32
7.4.2 Yleinen dialogi . . . 33
7.4.3 Erikoistuneet dialogit . . . 33
7.5 Palvelin . . . 34
7.5.1 WLAN . . . 34
7.5.2 Puhesyntetisaattori . . . 35

8 Testaus **36**

8.1 Testitapaukset 36

8.2 Tulokset 38

9 Pohdinta **40**

10 Johtopäätökset **42**

Lähteet **43**

Symboliluettelo

ADSL	Asymmetric Digital Subscriber line
ARIA	ActivMedia Robotics Interface for Application
DS	Direct Sequence
FH	Frequency Hopping
GPL	Gnu General Public License
GTK	Gimp Toolkit
GTKMM	Gimp Toolkit minus minus
HRI	Human-Robot-Interaction
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
LGPL	Library General Public License
MAC	Media Access Control
MVC	Model-View-Controller
PAC	Presentation-Abstraction-Control
PDF	Portable Document Format
PTZ	Pan-Tilt-Zoom
QoS	Quality of Service
ROV	Remote operated vehicle
RRT	Rapidly Exploring Random Tree
RTF	Rich Text Format
SONAR	Sound navigation and ranging
STL	Standard Template Library
TCP/IP	Transmission Control Protocol / Internet Protocol
UAV	Unmanned Air Vehicle
UDP	User Datagram Protocol
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

1 Johdanto

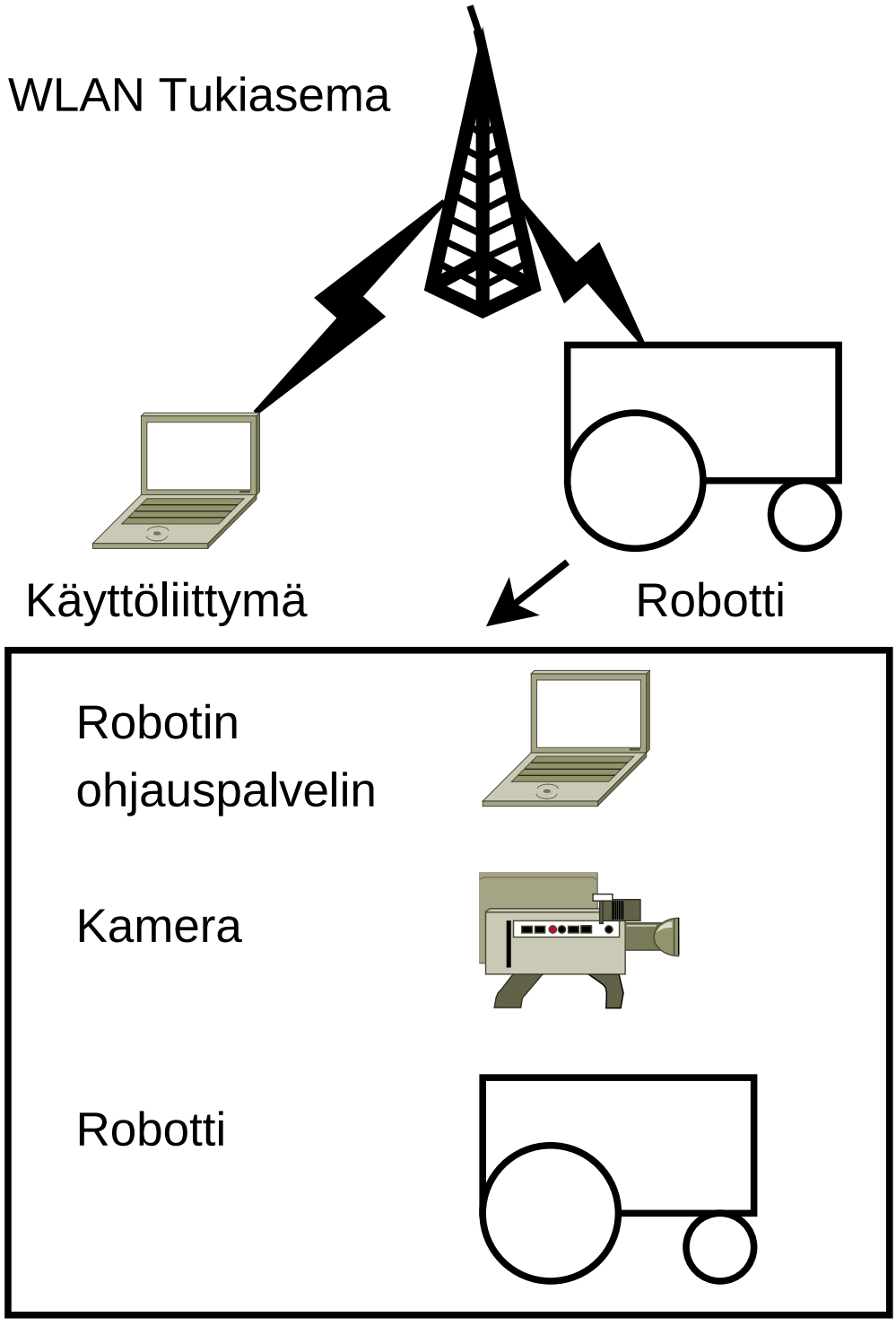
Autonomisille liikkuville roboteille löytyy jo monia käytännön sovelluksia, esimerkkinä tavaroiden siirtely varastossa ja itsenäisesti talossa liikkuva pölynimuri. Autonomialla ei ratkaista kuitenkaan kaikkia ongelmia, vaan myös kaukokäytölle ihmisen ohjauksessa on omat tarpeensa. Kaukokäytöllä tarkoitetaan robotiikassa ohjausta etäältä, langattoman tai langallisen yhteyden yli. Tavoitteena on pystyä ohjaamaan robottia tarkkailematta sitä paikan päällä. Käytännön sovelluksia kaukokäytölle löytyy esimerkiksi ihmiselle mahdollisesti haitallisista tehtävistä, kuten vaarallisien aineiden tai esineiden käsittelystä tai muuten toimimisesta ihmiselle vaarallisessa ympäristössä.

Lappeenrannan teknillisen yliopiston Tietotekniikan osaston Tietojenkäsittelytekniikan laitoksella on tutkimuskäytössä MobileRobots-yhtiön (entinen ActivMedia Robotics) valmistama pyörillä liikkuva robotti, jota voidaan ohjata langattoman lähiverkon (WLAN, wireless local area network) tarjoamien yhteyksien yli. Robotissa on ympäristön muotojen, etupäässä esteiden etäisyyksien, mittausta varten kaikuluotaimia. Mukana kulkee robotin toimintaa ohjaava sylimikro, joka on kytketty USB:n (universal serial bus) avulla robottiin. Mukaan voidaan myös liittää ohjattava kamera. Järjestelmä on esitetty karkealla tasolla kuvassa 1.

Robotin kehitystyökaluihin kuuluu ohjelmointikirjasto, jolla voidaan luoda asiakaspalvelin (client-server) -mallin mukaisia robotin kaukokäyttösovelluksia TCP/IP-yhteyksien (transmission control protocol / internet protocol) yli. Ohjelmointikirjasto kapseloi robotin käyttöön liittyvää logiikkaa ja tiedon varastointia, kuten esimerkiksi paikanmäärityksen robotin pyörien liikkeen perusteella ja kaikuluotaimien tietojen tallennuksen.

Peruskäyttöliittymä robotin kaukokäyttöön on MobileRobots-yhtiön vapaasti jakama MobileEyes-sovellus, joka sisältää muun muassa tuen robotin ohjauspalvelimelta tulevan kuvan näyttämiseen. Tämä peruskäyttöliittymä ei kuitenkaan ole laajennettavissa, koska se ei ole saatavilla lähdekoodina, eikä se tarjoa omaa ohjelmointirajapintaansa tai mitään muutakaan liittymää ulospäin. Robotin ohjauspalvelimesta puolestaan on saatavilla toimivien esimerkkitoteutuksien lähdekoodi.

Työn tavoitteena on luoda käyttöliittymäsovellus, joka tukee mahdollisimman kattavasti samoja ominaisuuksia kuin olemassaoleva MobileEyes-sovellus. Tärkeä tavoite on saada ohjelmiston arkkitehtuuri riittävän yksinkertaiseksi, mutta tarpeeksi jous-



Kuva 1: Yleiskuva robotin käyttöympäristöstä. Alempana laatikossa robotin mukana kulkeva kokonaisuus.

tavaksi, jotta uusien ominaisuuksien suunnittelu ja ohjelmointi olisi jatkossa sujuvaa. Työssä toteutetaan lisäksi kaksi lisäominaisuutta: WLAN-yhteyden tilan näyttö ja puhesyntetisaattorin ohjaus. Tämä vaatii työtä myös ohjauspalvelimen puolelle, ja myös palvelimeen liittyvä lähdekoodi halutaan helposti uudelleenkäytettäväksi.

Luvussa 2 esitellään kaukokäytön periaatteita, sovelluksi, sekä kaukokäyttöliittymiä. Lisäksi esitellään lyhyesti odometristä paikannusta, kaikuluotauksen toimintaa ja reitinsuunnittelun periaatteita. Luvussa 3 esitellään langattoman lähiverkon toiminta ja näkökohtia kaukokäytön kannalta. Luvussa 4 esitellään kaksi arkkitehtuuriratkaisua graafisillekäyttöliittymille. Luvussa 5 esitetään vaatimukset tehtävälle ohjelmistolle. Luvussa 6 kuvataan ohjelmiston arkkitehtuuri. Luvussa 7 esitellään käytännön toteutus. Luvussa 8 esitetään ohjelmiston testaussuunnitelma ja tulokset. Luvussa 9 pohditaan toteutuksen onnistumista ja jatkokehitystä. Viimeiseksi esitetään johtopäätökset työstä luvussa 10.

2 Robotiikkaa

2.1 Kaukokäytön periaatteet

Liikkuvan robotin kaukokäytössä on tavoitteena saada ohjattua robottia tarkasti ja saada mallinnettua ympäristö, jossa toimitaan. Tarkoitus on saavuttaa luotettava ohjaus ja yleiskuva, jotta robottia voidaan käyttää tehtävien suoritukseen tarkkailematta sitä paikan päällä. Tämä erottaa sen arkipäiväisemmästä etäohjauksesta (remote control). Kaukokäyttö edellyttää robottijärjestelmältä

- monipuolista ohjattavuutta,
- luotettavaa navigointia (kartat, paikannus, ...) ja
- sensoridatan saamista ympäristöstä (kaikuluotaus, video, ...) [1] [2].

Karkealla tasolla kaukokäyttöä voidaan jakaa autonomisen toiminnallisuuden perusteella [1] [2]. **Suorassa ohjauksessa** (direct control) säädetään suoraan ohjausparametreja, ilman korkeamman tason toiminnallisuutta. Esim. robotin tarttumakäden kulman asento kerrotaan absoluuttisina asteina tai astemuutoksena. Ohjaaja säätää jokaista mahdollista parametria tarkasti.

Jaettu ohjaus (shared control) tarkoittaa ohjausvastuun jakamista käyttäjän ja robotin kesken. Esimerkiksi työn kohteena olevassa järjestelmässä robotti osaa pysähtyä havaittuaan esteen, ja reitinsuunnittelufunktiolta voidaan pyytää karttaan perustuen reitti, jonka robotti kulkee itsenäiseti. Robotin kulkiessa reittiä voidaan keskittyä kameran ohjaamiseen ja kuvan tarkkailuun. Läheisesti liittyvä käsite on **vaihdeltu ohjaus** (traded control), jossa ohjausvastuuta jaetaan ajallisesti, esim. robotti suunnittelee reitin ensin tiettyyn pisteeseen josta operaattori jatkaa suoralla ohjauksella.

Korkean tason ohjaus (supervisor control) pitää periaatteena antaa robotille korkean tason käsky, jonka suoritusta ihminen jää tarkkailemaan. Robotin on suotavaa antaa hoitaa tehtävää autonomisesti, kun suoran ohjauksen käyttäminen on vaikeaa, esimerkiksi avaruussovelluksissa.

Joissain tapauksissa, kuten vanhemmissa tutkimuksissa, kaukokäyttö määritellään tiukasti suoraksi ohjaukseksi. Kaukokäytön sisältävä yleisempi käsite HRI (human-

robot interaction) sisältää autonomisuuden eri asteet ja erilaisien ohjausmenetelmien ja käyttöliittymien tarkastelun [3].

2.2 Kaukokäytön sovellukset

Maalla liikkuvia kaukokäytettyjä ajoneuvoja käytetään tutkimiseen, seurantaan ja tiedusteluun. Näihin voi liittyä merkittäviäkin vaaratekijöitä, kuten tuhoutumisen mahdollisuus. Kategoriaan kuuluvat myös avaruussovellukset, kuten Lunokhod Neuvostoliiton ajalta ja moderni Mars Sojourner [1]. Kaivostoiminnassa käytetään robotteja esimerkiksi katosta tukemattomien tunneleiden tutkimiseen [4]. Toinen vaarallisen työn kohde on erilaisten räjähteiden kuten miinojen etsiminen [5].

Lentäviä miehittämättömiä aluksia (UAV, unmanned air vehicle) käytetään sotilaskäytössä tiedusteluun ja kohteen tunnistamiseen [1].

Vedenalaiset alukset ovat merkittävä sovelluskohde. Yleisnimenä näille on ROV (remote operated vehicle). Käyttökohteita ovat esimerkiksi merenpohjan kartoitus, vedessä olevien rakenteiden kunnon tarkkkailu ja esineiden nouto merenpohjasta [1].

Lääketieteessä ensimmäisiä laajimpia robotiikan sovelluskohteita olivat ortopediaan (luukirurgia ja kudosvammat) ja laparoskopiaan (vatsaontelon tähystys) liittyvät kirurgiset toimenpiteet. Näissä hyödynnetään robottien tarjoamaa tarkuutta ja toistettavuutta. Robotit tarjoavat myös mahdollisuuden mahdollisimman pienen jäljen jättämiseen [6]. ADSL-verkon (asymmetric digital subscriber line) yli on onnistuneesti suoritettu stereotaktinen (kolmiulotteisesti kuvattu) leikkaus aivoverenvuotopotilaalle [7].

2.3 Kaukokäyttöliittymät

Kattava kuva ympäristöstä ja monipuoliset ohjaus- ja navigointimahdollisuudet tarkoittavat suurta tiedonvälityksen määrää. Käyttöliittymissä pyritään saamaan mahdollisimman paljon tietoa esille, kuitenkaan ylittämättä käyttäjän henkistä ja fyysistä kapasiteettiä. Robotin autonominen toiminta auttaa käyttäjän työkuorman keventämisessä, mutta informaatio robotin toiminnasta halutaan silti saada selville tarkasti. Lisäksi suuren tietomäärän välitys asettaa vaatimuksia robotin ja kaukokäyttöliittymän siirtotien välille [1] [2]. Nykyisissä liikkuvien robottien kaukokäyttöliittymissä onkin usein ongelmia käytettävyydessä, informaatiopaljouden hallinnassa ja robotin

ympäristön hahmottamisessa [8].

Eräitä keinoja käyttäjän kapasiteetin hallitsemiseen ovat haptiset (kosketusaistiin perustuvilla) käyttöliittymän osat ja laajennetun todellisuuden (augmented reality) sovellukset. Molemmilla pyritään ottamaan käyttäjän luonnollisia aisteja kokonaisvaltaisemmin käyttöön. Esimerkiksi haptisessa tapauksessa robotin ohjaussauva voi vastustaa liikettä esteen suuntaan. Laajennetussa todellisuudessa voidaan esimerkiksi luoda stereokuva ympäristöstä, joka on ihmiselle luonnolisempi etäisyyksien hahmottamiseen [9] [10].

Tutkimuksessa [11] selvitettiin kaukokäyttöliittymän ulkoasun vaikutusta liikkuvalla robotilla työskennellessä. Lentokoneen ohjaamon kaltainen näyttö mahdollisti nopeimman työskentelun sekä kokeneille, että aloitteleville robotin ohjaajille. Silti käyttäjille mieluisampi oli Windows-ikkunointimallin mukainen käyttöliittymä, jossa jokainen informaatio oli omassa ikkunassaan ikkunointijärjestelmän hoidettavana. Ikkunoiden kokoa ja paikkaa pystyi vapaasti säätämään. Työn tehokkaan suorittamisen kannalta käyttöliittymän visuaalinen hienosäätö ei ollut otosryhmälle oleellista, vaan painopiste on selkeässä ja saatavilla olevasta kaukokäyttötiedosta. Käyttömukavuuden kannalta taas natiivin ikkunointijärjestelmän tyylinen käyttöliittymä koettiin miellyttävämpänä.

Viive robottia komennettaessa ja palautteen saamisessa vaikeuttaa tehtävän suorittamista. Ihmisen reagointiaika visuaaliseen tai tuntoärsykkeeseen on muutaman kymmenen millisekunnin luokkaa, esimerkiksi elokuvien 25 kuvaa/sekunti tarkoittaa 40ms/kuva, jolloin kuvien vaihtuminen ei enää erotu. Tutkimuksessa [12] tehtävän suoritusajan havaittiin kasvavan melko lineaarisesti viiveen kasvaessa. Kokonaissuoritusaika ei enää juuri kasva sekunnin luokkaa olevan viiveen jälkeen. Viiveen arvot 0.6 ja 1 sekuntia ovat rajapyykkejä käyttäjän strategian vaihtumsille.

2.4 Odometrinen paikannus

Käytetty robotti käyttää liikkumiseen ja ohjaamiseen differentiaalipyörästöä. Molemmat vetävät renkaat liikkuvat itsenäisesti, joten robotti voi pyöriä paikallaan. Paikanmääritys tehdään mittaamalla akselien pyörähdystä, ja laskemalla tästä renkaiden itsenäisesti kulkema matka. Odometrinen paikannus on toteutettu valmiiksi ARIA-kirjastossa (ActivMedia Robotics Interface for Application).

Systemaattisia, kalibroinnilla hallittavia virheitä ovat:

- renkaiden halkaisijoiden vaihtelut,
- renkaiden kohdistusvirhe,
- akselin pyörähdyskulman mittauksen resoluutiorajoitus,
- akselin pyörähdyskulman mittauksen taajuusrajoitus.

Ei-systemaattisia virheitä aiheuttavat:

- epätasainen maasto,
- renkaiden pidon murtuminen.

UMBmark-menetelmällä [13] voidaan määrittää virheherkkyys systemaattiselle virheelle, ja myös arvoida karkeasti herkkyyttä ei-systemaattisille virheille.

2.5 Kaikuluotaus

Kaikuluotauksella (sonar, sound navigation and ranging) tehtävä etäisyydenmittaus perustuu tyypillisesti 40-180kHz ultraäänipurskeiden heijastumisajan mittaukseen kohteen pinnasta. Roboteissa käytettävien kaikuluotaimien mittausalue on noin 12cm-5m. Suurempien etäisyyksien mittaaminen rajoittaa mittauksien määrää aikayksikköä kohden. Kaikuluotainten tarkkuus on robottisovelluksissa luokkaa 98% - 99.1 %. Matalammat ultraäänitaajuudet kasvattavat maksimietäisyyttä, mutta myös minimietäisyyttä [14]. ARIA-kirjasto käsittelee ja varastoi sensoridatan, joka on suoraan saatavilla xy-koordinaatteina ohjausserverin kartan suhteen.

2.6 Reitinsuunnittelu

Käytetty kartta jaetaan tavallisimmin osiin reitin määrittämiseksi [14]:

Tiekartta (road map). Haetaan mahdolliset reitit kartasta seuraten esteiden reunoja.

Soluhajotus (cell decomposotion). Erotetaan karttasolut, soluun voi kulkea tai ei. Kulkukelpoisien solujen välille luodaan graafi, josta määritellään kuljettava reitti.

Potentiaalikenttä (potential field). Luodaan matemaattisesti matala potentiaali kohteeseen, jota päin robottia ohjataan. Esteet näkyvät jyrkän korkeina potentiaaleina.

Reitti voidaan myös hakea osin satunnaisesti, kuten esimerkiksi RRT-tekniikalla (rapidly-exploring random tree) [15]. Etuna on menetelmän potentiaalinen nopeus, mutta haittana epädeterministisyys.

Työssä käytetty MobileRobots-yhtiön erillinen ARNL-kirjasto toteuttaa reitinhaun. Kirjasto käyttää kartan ruudukkopohjaista (grid-based) suunnittelua, eli kyseessä on soluhajotusmenetelmä [16]. Kaikuluotauksella tai muulla ARIA-kirjastojen tukemalla tavalla havaitut esteet otetaan huomioon dynaamiseen ikkunaan pohjautuvan menetelmän avulla. Menetelmässä tarkistetaan joka aikayksikköä kohden uudelleen robotin lähiympäristö, ottaen huomioon sensorien ilmoittamat esteet. Lähiympäristö on alue, johon robotin on mahdollista päästä aikayksikössä, ottaen huomioon robotin nykyinen nopeus ja kääntyvyys, sekä näiden maksimimuutokset [17].

3 Langaton lähiverkko

3.1 Toiminta

Langattomalla lähiverkolla eli WLAN:illa, tarkoitetaan nykypäivänä pääasiassa IEEE standardi 802.11:n toteuttavien laitteiden radioteitse muodostamaa lähiverkkoa. Standardi määrittelee WLAN:in fyysisen siirtotien ja MAC-protokollan (media access control). Nämä vastaavat OSI-verkkomallin kahta alinta kerrosta, fyysistä ja linkkikerrorsta. TCP/IP on näiden yläpuolelle verkko- ja siirtokerrokselle sijoittuva protokollapari, jota WLAN:in päällä voidaan ajaa. Fyysiselle kerrokselle on määritelty infrapunaa käyttävä optinen standardi, ja hajaspektriä hyödyntävä radiostandardi. Hajaspektrillä tarkoitetaan kantoaallon hajautusta, jolla pyritään parantamaan signaali/kohinasuhdetta, saamaan usea lähetin/vastaanotinpari toimimaan samalla taajuusalueella, ja ehkäisemään radiosignaalin heijastumisista johtuvia ongelmia. Standardi määrittää kantoaallon moduloinnin taajuushyppelylle (frequency hopping, FH) tai suorasekvenssitekniikalle (direct sequence, DS). Taajuushyppelyssä kantoaallon taajuutta vaihdetaan nopeaan tahtiin, suorasekvenssissä kerrotaan data pseudosatunnaislukusarjalla. Radiostandardi käyttää 2.4GHz taajuusaluetta ja nimellinen siirtonopeus on 1 tai 2 megabittiä sekunnissa [18] [19]. Uudemmat standardit 802.11e ja 802.11g tukevat vastaavasti liikenteen laatuluokitusta (quality of service, QoS) ja suurempia kaistanleveyksiä [20].

Verkkotopologioita on kahdenlaisia, ad hoc-verkkoja ja infrastruktuureja. Verkon laitteet voidaan niinikään jakaa kahteen tyyppiin, mobiiliasemiin (mobile station) ja tukiasemiin (access point). Ad hoc-verkoissa mobiiliasemat ottavat suoraan yhteyttä toisiinsa, kun taas infrastruktuurissa mobiiliasemat liikennöivät tukiasemien kautta. Tukiasema voi jälkimmäisessä tapauksessa tarjota yhteyden langalliseen lähiverkkoon.

IP-verkoissa on käytössä yhteydellinen TCP tai yhdeydetön datagrammiprotokolla UDP (user datagram protocol). TCP takaa datan saapumisen ja säilyttää datapakettien järjestyksen. UDP ei takaa datapakettien perillemenoa eikä järjestystä. Yksinkertaisempana protokollana UDP on käytössä reaaliaikasovelluksissa nopeutensa takia. TCP hoitaa vahvistuksen ja datan järjestämisen, mutta viiveet voivat vaihdella enemmän, joka on ongelmallista reaaliaikasovelluksissa [21].

ARIA-kirjasto muodostaa sovellusrungossa asiakkaalta palvelimeen TCP-yhteyden. Palvelimelta asiakassovellukseen on käytössä myös UDP-yhteys. Palvelin voi halu-

tessaan lähettää tietoa joko TCP- tai UDP-yhteyttä pitkin.

3.2 WLAN kaukokäytössä

WLAN-komponenttien hinta, yleisyys, kompaktius, sekä TCP/IP:n toimiminen suoraan yleisimmissä käyttöjärjestelmissä WLAN:in yli nopeuttaa käyttöönottoa ja kehitystä. Rajoitukset tulevat vastaan yhteyden maksimietäisyyksissä, latensseista ja tiedonsiirtoon käytettävissä olevasta kaistasta.

Maksimietäisyys yhteyksille on kilometrinkin luokkaa esteettömässä tilassa, mutta laitevalmistajien ilmoittama maksimietäisyys yhteydelle rakennuksen sisällä on 100-250 metrin luokkaa [18]. Tätäkään ei aina saavuteta, sillä rakennuksen muodoista ja materiaalista johtuen saattaa rakennuksen sisälle muodostua katvealueita.

Pienillä kuormilla WLAN-yhteyksien latenssit ovat muutaman millisekunnin luokkaa. Suurimat latenssit ovat noin 70-80 millisekunnin luokkaa, ennenkuin paketteja alkaa hävitä kaistan ollessa kokonaan käytössä. Siirrettäessä kuvaa 802.11b-standardin 2 Mbit/s rajoitus voi tulla äkkiä vastaan, esimerkiksi 160x120 YUV 4:1:1 formaatti 10 kuvaa sekunnissa vaatii noin 2.2 Mbit/s ilman tiedon pakkausta [20].

4 Graafiset käyttöliittymät

MVC- (Model - View - Controller) [22] [23] ja PAC-arkkitehtuurit [24] (Presentation - Abstraction Control) ovat graafisten käyttöliittymäohjelmistojen (GUI, graphical user interface) toteutusperiaatteita. Kuvassa 2 on esitetty arkkitehtuurien komponentit. MVC-arkkitehtuuri jakaantuu osiin seuraavasti:

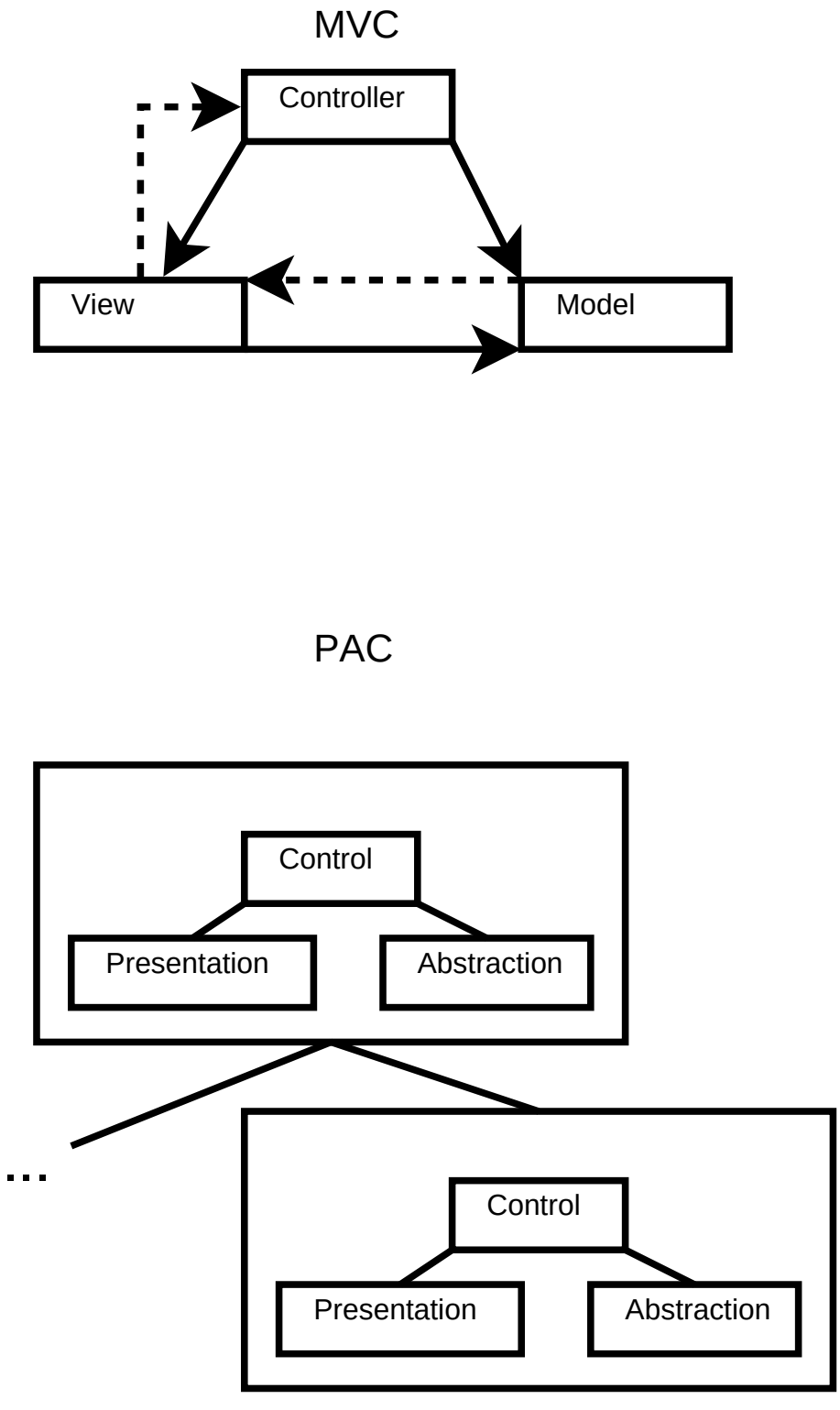
Mallissa (model) pidetään tieto sovelluskohtaisesta tiedosta. Malli voi esimerkiksi kapseloida tietokantaliittymän, hakien ja päivittäen kannan sisältöä. Malli ilmoittaa datan päivittymisestä näkymälle.

Näkymä (view) on käyttäjälle näkyvä komponentti. Näkymä hakee mallin niin ilmoittaessa tietoa visualisoitavaksi. Ohjaimelle kerrotaan käyttäjän syötteet.

Ohjain (controller) ottaa vastaan käyttäjän komentoja (tai muuta syötettä), ja muokkaa niiden perusteella mallissa olevaa dataa mallin omilla metodeilla, sekä tarvittaessa päivittää myös näkymää.

PAC-arkkitehtuuri jakaa myös tiedon ja esityksen. Erona se, että presentation- ja abstraction-komponentit keskustelevat ainoastaan control-olion välityksellä, ja presentation-komponentti käsittelee käyttäjän syötteet. Lisäksi PAC-triadit järjestetään hierarkiaan. Triadit välittävät hierarkiassaan viestejä toisilleen control-olioidensa välityksellä. Yksi triadi käsittelee varsin pientä osaa koko järjestelmästä.

MVC-arkkitehtuuri on yksinkertaisempiin järjestelmiin helpompi toteuttaa, koska PAC-arkkitehtuurissa toiminnallisuutta on jaettava pienempiin osiin triadihierarkiaksi. Suurissa järjestelmissä hierarkiasta on vastaavasti etua ylläpidon kannalta. Vaikeutena MVC-arkkitehtuurin toteutuksessa perustason graafisilla työkalukirjastoilla (toolkit) on käyttäjän syötteiden käsittelyn erottaminen [25].



Kuva 2: MVC- ja PAC-arkkitehtuurit.

5 Vaatimukset

Työlle ei määritelty tarkkoja muodollisia vaatimuksia. Vaatimukset esitettiin yleisellä tasolla seuraavasti:

- Ohjelmiston toimiminen Linux-ympäristössä.
- Muokattavissa oleva arkkitehtuuri.
- Dokumentoitu koodi.
- MobileEyes-sovelluksen toiminnallisuuden toistaminen.
- Graafinen asetusten hallinta.
- Uusi ominaisuus: WLAN-yhteyden tilan näyttäminen käyttöliittymässä.
- Uusi ominaisuus: puhesyntetisaattorin ohjaus.

5.1 Linux-ympäristö

Ohjelmiston tulee kääntyä ja toimia jossain yleisessä Linux-jakelussa. Työssä kehityskäytössä oli Debian-jakelun versio "sarge" [26]. Muita käyttöjärjestelmiä, kuten Microsoft Windows, ei tarvitse tukea.

Käytettävien työkalujen tulee olla riittävän yleisesti saatavilla. Työkalut ja ohjelmointikirjastot voivat olla esim. GPL:n (Gnu General Public License) [27] alaisia. Vaikka lisenssi voi pakottaa ohjelmiston lähdekoodin avoimeksi, tämän ei katsota olevan liian rajoittava tekijä, sillä ohjelmisto tehdään tutkimuskäyttöön.

Ohjelmisto kehitetään C++-kielellä, koska robotin oma kirjasto on myös samalla kielellä kirjoitettu. Käytetty kääntäjä on GCC (Gnu Compiler Collection).

Ohjelmiston graafinen osuus kehitetään yleisen toolkit-tason kirjaston pohjalle. Toteutukseen valittiin pohjaksi GTKMM-kirjasto [28], joka on virallinen GTK+-kirjaston [29] liittymä C++-kielelle. Korkeamman tason graafinen kehitysympäristö arvioitiin olevan liian arka versiomuutoksille suhteessa tuotettuun koodiin ja käyttöjärjestelmään.

Käyttöliittymän ulkonäkö suunnitellaan Glade-ohjelmalla [30], joka tuottaa perustan luokkarakenteelle C++-koodina käyttäen GTKMM-kirjastoa.

Ohjelmistoa pidetään kehityksen ajan versionhallinnassa CVS:n [31] avulla tietotekniikan osaston palvelimella.

5.2 Arkkitehtuuri

Käyttöliittymän arkkitehtuuri tulee olla riittävän selkeä jatkokehitystä ajatellen. Vaatimuksen toteutumista on vaikea todeta, mutta jotain odotuksia siitä voidaan vetää.

- Toiminnallisuutta tulee jakaa osiin.
- Käyttöliittymää suoraan käsittelevä toiminnallisuus ei saa olla tiukasti kytketty robotin käyttöön liittyvään toiminnallisuuteen.
- Toiminnallisuutta on pystyttävä laajentamaan.

5.3 Dokumentaatio

Ohjelmistosta on tuotettava dokumentaatiota jatkokehitystä varten. Tämä tarkoittaa vähintään tärkeimpien luokkien ja metodien kommentointia lähdekoodiin. Tärkeimmät luokat ovat korkeimman tason olioita, jotka käsittelevät itse robottia tai käyttöliittymää.

Suositeltu työkalu dokumentaation tuottamiseen on doxygen, jolla voidaan parsia määrämuotoon kirjoitettu informaatio lähdekoodista, ja tuottaa siitä yhtenäinen dokumentaatio useisiin eri muotoihin kuten HTML (hypertext markup language, RTF (rich text format) tai PDF (portable document format) [32].

5.4 MobileEyes-sovelluksen toiminnallisuus

Käyttöliittymän tulee hakea ja näyttää seuraavaa perustietoa robotista:

- status,
- toimintamoodi,
- akun varaus,
- x- ja y-positio kartalla,

- suunta,
- kulkunopeus,
- kääntymisen nopeus.

Käyttöliittymässä on oltava karttanäyttö, johon kuvataan robotti ja sen kulkusuunta. Karttanäyttöä käytetään myös kaikuluotauksen tiedon, kartan ja reitin visualisoimiseen.

Käyttöliittymän on osattava komentaa robottia liikkumaan eteen ja taakse, sekä kääntymään haluttuun suuntaan. Ohjaus tapahtuu nuolinäppäimillä tai graafisilla painikkeilla.

Käyttöliittymästä on voitava kytkeä pois ja päälle turvamekanismi, joka estää ajamisen, mikäli kaikuluotain havaitse esteen liian lähellä kulkusuunnassa.

Käyttöliittymästä on voitava aktivoida vaellusmoodi, jossa robotti kulkee autonomisesti, pysähtyen esteen kohdatessaan ja ottamalla uuden suunnan.

Käyttöliittymästä tulee voida antaa pysähtymiskomento.

Ohjauspalvelin lähettää kaikuluotauksella mitattuja etäisyyksiä, jotka käyttöliittymän tulee näyttää karttanäytöllään.

Käyttöliittymän tulee osata pyytää ohjauspalvelimelta kameran kuvaa. Saatu kuva on näytettävä käyttöliittymässä. Lisäksi kameraa tulee voida ohjata.

Ohjauspalvelimella voi olla saatavilla karttatiedosto, jonka käyttöliittymä voi pyytää näyttämistä varten. Käyttöliittymään on toteutettava toiminnallisuus joka lataa palvelimella mahdollisesti olevan kartan ja näyttää sen robotin karttanäytön pohjalla. Käyttöliittymän tulee osata liikuttaa karttaa robotin liikkeen mukaan.

Käyttöliittymän on osattava kertoa ohjauspalvelimen reitinmäärityksselle haluttu loppupiste, ja näytettävä reitinmäärityksen laskema reitti kartalla.

5.5 Graafinen asetusten hallinta

Käyttöliittymän tulee tarjota graafinen työkalu ohjelman asetusten hallintaan. Tavoitteena on luona yleisesti käytettävä toiminnallisuus. Esimerkkinä pyritään hallitsemaan tällä toiminnallisuudella viiveet, jotka ilmaisevat miten usein asiakaskäyt-

töliittymä käy hakemassa tietoja ohjauspalvelimelta. Lisäksi asetukset tulee voida tallettaa tiedostoon.

5.6 WLAN-yhteyden tila

Robotin ohjauspalvelimen tulee hakea käyttöjärjestelmältä tiedot käytetyn WLAN-yhteyden tilasta. Asiakaskäyttöliittymän tulee osata hakea tämä tieto ja näyttää se.

5.7 Puhesyntetisaattorin ohjaus

Käyttöliittymän tulee tarjota mahdollisuus syöttää puheeksi syntetisoitavaa tekstiä. Tämä teksti lähetetään ohjauspalvelimelle, jossa tekstistä syntetisoidaan puhetta. Tarkoitus on käyttää MobileRobots-yhtiön ArFestival-puhesyntetisaattoria.

6 Arkkitehtuuri

6.1 ARIA-kirjasto

Viestinvälitykselle käyttöliittymän ja ohjauspalvelimen välille luo pohjan ARIA-kirjasto. Työn kannalta arkkitehtuuriin vaikuttava osuus on viestejä vastaanottavien funktorien rekisteröinti ARIA-kirjastosta periytetylle asiakas- tai palvelinolion rungolle. Saadessaan viestin verkosta, ARIA-kirjasto kutsuu funktio-oliota eli funktoria antaen parametriksi saadun paketin. Funktori on "tyhjä metodi", johon sovelluksen täytyy asettaa varsinainen toiminnallisuuden tarjoama metodi. Funktorissa puolestaan kutsutaan asetettua metodia. Viestit tunnistetaan paketteihin asetettavan selväkielisen merkkijonon avulla. Käsittelijä viestipaketille saadaan käyttöön kahdella toimenpiteellä:

1. Luodaan funktori luokkamallista (class template) ja alustetaan se viitteellä käsittelijämetodiin.
2. Rekisteröidään funktori runko-olioon. Rekisteröinnissä annetaan paketin tunnisteena toimiva merkkijono.

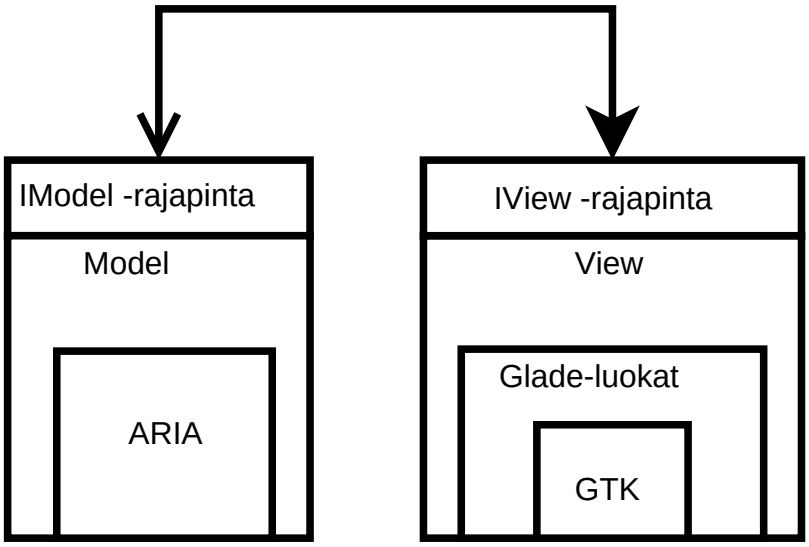
Lähetys vastapään oliolle suoritetaan luomalla ARIA-kirjaston pakettia kuvaava olio ja kutsumalla tälle runko-olion lähetysmetodia. Asiakasolio voi myös pyytää lähettämään uusia vastauksia tiettyyn pyyntöön määrätyin välijajoin ilman uutta pyyntöä.

6.2 Malli-näkymä

Arkkitehtuurilliseksi lähtökohdaksi otettiin visuaalisen komponentin erottaminen varsinaisesta robotin ohjaus- ja tiedonkäsittelylogiikasta. MVC-arkkitehtuuria yksinkertaistetaan siten, että ohjainkerrosta ei ole. Karkean tason arkkitehtuuri koostuu vain kahdesta komponentista, mallista ja näkymästä. Arkkitehtuuri pääluokkineen ja kirjastoineen on esitetty kuvassa 3.

Säikeistys (threading) nähdään välttämättömänä ARIA- ja GTKMM-kirjastojen tapahtumankäsittelysilmukoiden (event loop) yhdistämiseen, joten arkkitehtuurillisesti ajatellaan mallia ja näkymää ajettavan omissa säikeissään.

Malli ja näkymä voivat molemmat käyttää ARIA-kirjaston tarjoamaa toiminnallisuutta. Malli ei saa käyttää GTKMM-kirjastoa, tai muita näytön toteutuksessa tar-



Kuva 3: Käyttöliittymän arkkitehtuuri.

vittavia kirjastoja. Järjestelyn tarkoituksena on pitää malli mahdollisimman riippumattomana näkymästä, jolloin näkymäkomponentin mahdollinen korvaaminen tulevaisuudessa on helpompaa.

6.2.1 Rajapinnat

Luokilla tulee olemaan osoitin toisiinsa, ja ne tulevat välittämään tietoa toisilleen vain rajapintojensa kautta. Kuvassa 3 nämä ovat rajapinnat `IModel` ja `IView`. Näkymän rajapinnassa ei ole metodeja, joilla voisi suoraan muokata tietyn graafisen komponentin sisältöä. Mallin on kutsuttava näkymän rajapinnan metodia, joka ilmoittaa uuden tiedon saatavuudesta. Näkymä saa näin tiedon päivittyvän tiedon tyypistä ja hakee mallista halutuntyyppisen tiedon, jonka jälkeen se voidaan visualisoida. Käytännössä malli tulee tietämään vain yhden metodin näkymästä. Tällä on tarkoituksena saada malli mahdollisimman riippumattomaksi näkymästä MVC-arkkitehtuurin mukaisesti. Malli puolestaan pyrkii tarjoamaan kattavan metodivalikoiman robotin ohjaukseen ja tiedonsaantiin.

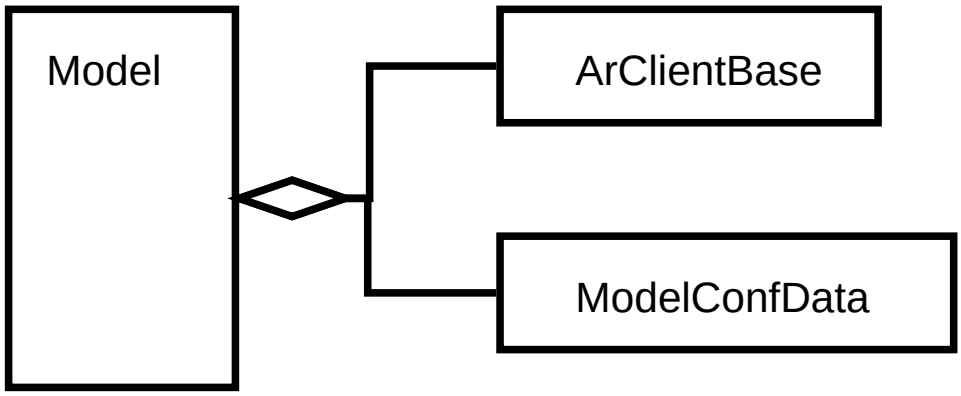
6.2.2 Malli

Mallin toteuttaa luokka `Model`, joka toimii adapterina [33] (adapter, wrapper) `ArClientBase`-luokalle. joka on ARIA-kirjaston asiakassovellukseen perustoiminnal-

lisuuden tarjoava luokka. `Model`-luokkaa ei periytetä, vaan se luo `ArClientBase`-olion sisäisesti käyttöönsä alustuksen yhteydessä. Lisäksi malliin on yhdistetty asetusten hallinta luokalla `ModelConfData`, jonka malli myös luo alustuksen yhteydessä. Mallin pääluokkarakenne on esitetty kuvassa 4.

Mallin rajapinta tarjoaa metodeja robotin ja kameran ohjaamiseen seuraavin käskyin:

- liikkuminen,
- pysähtyminen.
- videon haku,
- kameran ohjaus,
- vaellusmoodin aktivointi,
- turvallisen ajon aktivointi,
- loppupisteen asetus reitinsuunnittelulle ja
- puhesyntetisaattorin ohjaus.



Kuva 4: Mallin luokkarakenne.

Mallille komentoja antavat metodit nimetään prefiksillä `req_<toiminto>`. Nämä metodit aiheuttavat pyyntöviestin muodostamisen ja lähetyksen palvelinoliolle.

Robotin tietojen hakemista varten tulee mallin rajapinnan tarjota metodeja, jotka ovat säieturvallisia (thread safe). Säieturvallisuuden saavuttamiseksi on tarkoitus

luoda yleispätevä ratkaisu, jota voidaan soveltaa jokaiselle haettavalle tiedolle. Ratkaisun tulee kapseloida haettu data ja siihen liittyvä mutex (mutual exclusion lock), jolla säie lukitsee suorituksen itselleen tiedonhaun ajaksi. Haettavaa tietoa ovat

- xy-positio kartalla + muut perustiedot,
- kaikuluotaukset,
- video,
- käytetty pohjakartta,
- suunniteltu reitti,
- wlan-informaatio ja
- asetukset.

Mallilta tietoa hakevat metodit nimetään prefiksillä `get_<toiminto>`.

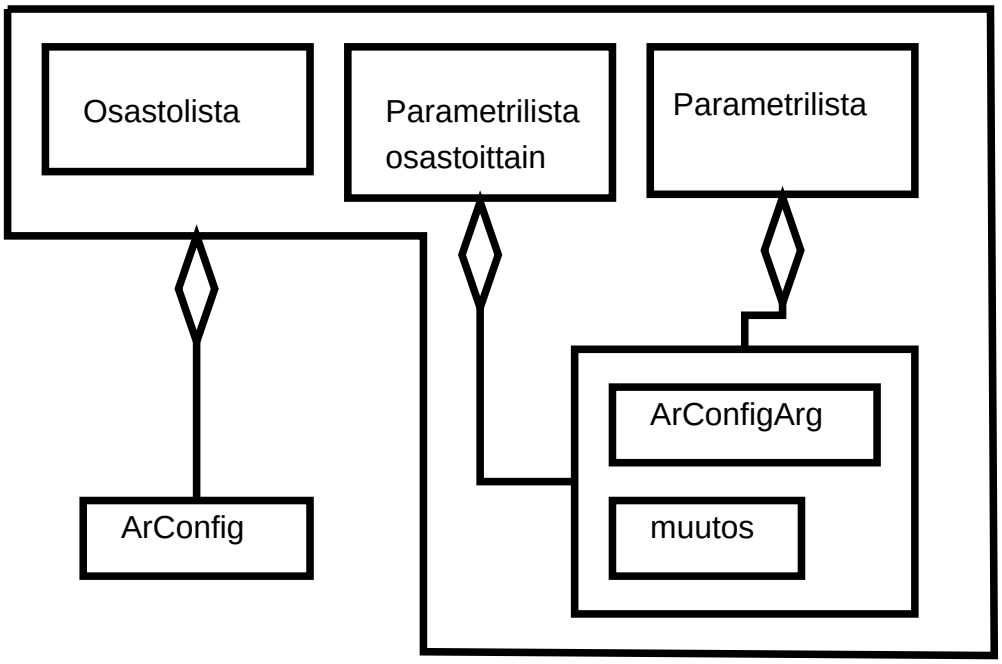
6.2.3 Asetusten hallinta

ARIA-kirjasto tarjoaa valmiin asetustenhallinnan, joka tukee asetusten jakamista osastoihin ja asetusten tiedostoon tallennusta. `ArConfig`-luokan olio kapseloidaan asetustenhallintaluokkaan. Yksittäiset asetukset kuvataan ARIA-kirjaston `ArConfig`-luokalla. Asetustenhallinta säilöö tietueita, joissa on osoitin `ArConfig`-olioon ja tieto, onko parametria modifioitu. Asetusta talletettaessa tallennetaan osastolistaan mahdollinen uusi osasto, tietue osastoittaiseen parametrilistaan ja tietue kaikki parametrit käsittävään listaan. Asetustenhallinnan luokkamalli on esitetty kuvassa 5.

Kapseloiva luokka `ModelConfData` tarjoaa metodit asetusten luomiseen, osastolistan hakemiseen, asetusten hakemiseen osastoittain, ja yksittäisen asetuksen hakemiseen. Malli tarjoaa metodin asetusten tiedostoon tallettamiseen.

6.2.4 Näkymä

Näytön luokkarakenne on esitetty kuvassa 6. Näkymän rungon muodostaa Glade-kehittimen luomasta sovelluksen perusluokasta periytetty luokka `View`. Käyttäjän syötteet kuten esimerkiksi napin painallus tai tekstikentän syötteen käsittely suoritetaan Glade-luokan metodissa, joka yliajetaan sisältämään haluttu toiminnallisuus



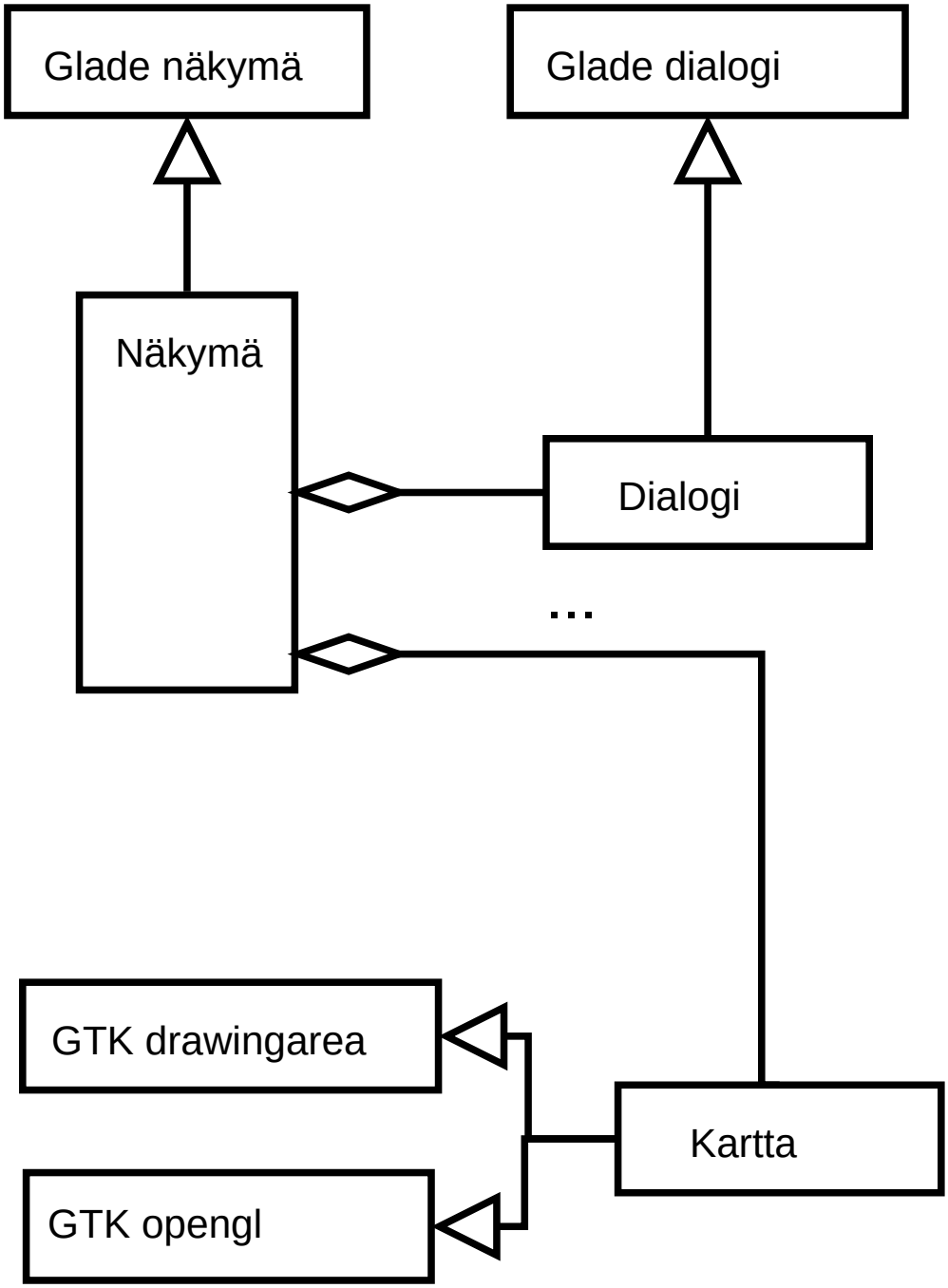
Kuva 5: Asetusten hallinnan luokkamalli.

varsinaisessa periytetyssä näkymäluokassa. Karkea hahmotelma käyttöliittymän ulkoasusta on esitetty kuvassa 7.

Näkymäluokan on luotava tarpeen mukaan dialogi-ikkunoita, jotka ovat myös Glade:lla hahmotelluista luokista periytettyjä. Dialogit ja näkymä keskustelevat ARIA-kirjaston funktoreiden tai näkymän julkisten metodien avulla. Dialogi-ikkunoiden käyttötarkoitukset ovat

- yhteyden muodostaminen palvelimeen,
- asetusten hallinta,
- törmäysten tarkistusten poiskytkentä ja
- ohjelmasta poistumisen varmistus.

Karttaikkunan toteuttamiseen tarvitaan erikoisluokka, johon voidaan piirtää grafiikkaa OpenGL-kirjaston [34] avulla. Tämä luokka periytetään GTKMM-kirjaston DrawingArea-luokasta ja gtkglextmm-kirjastosta saatavasta GL-luokasta. Karttaluokka saa osoittimet sekä malliin että näkymään. Mallilta haetaan näytettävä data ja koordinoidaan yhteistyö näkymän kanssa.



Kuva 6: Näkymän luokkamalli.

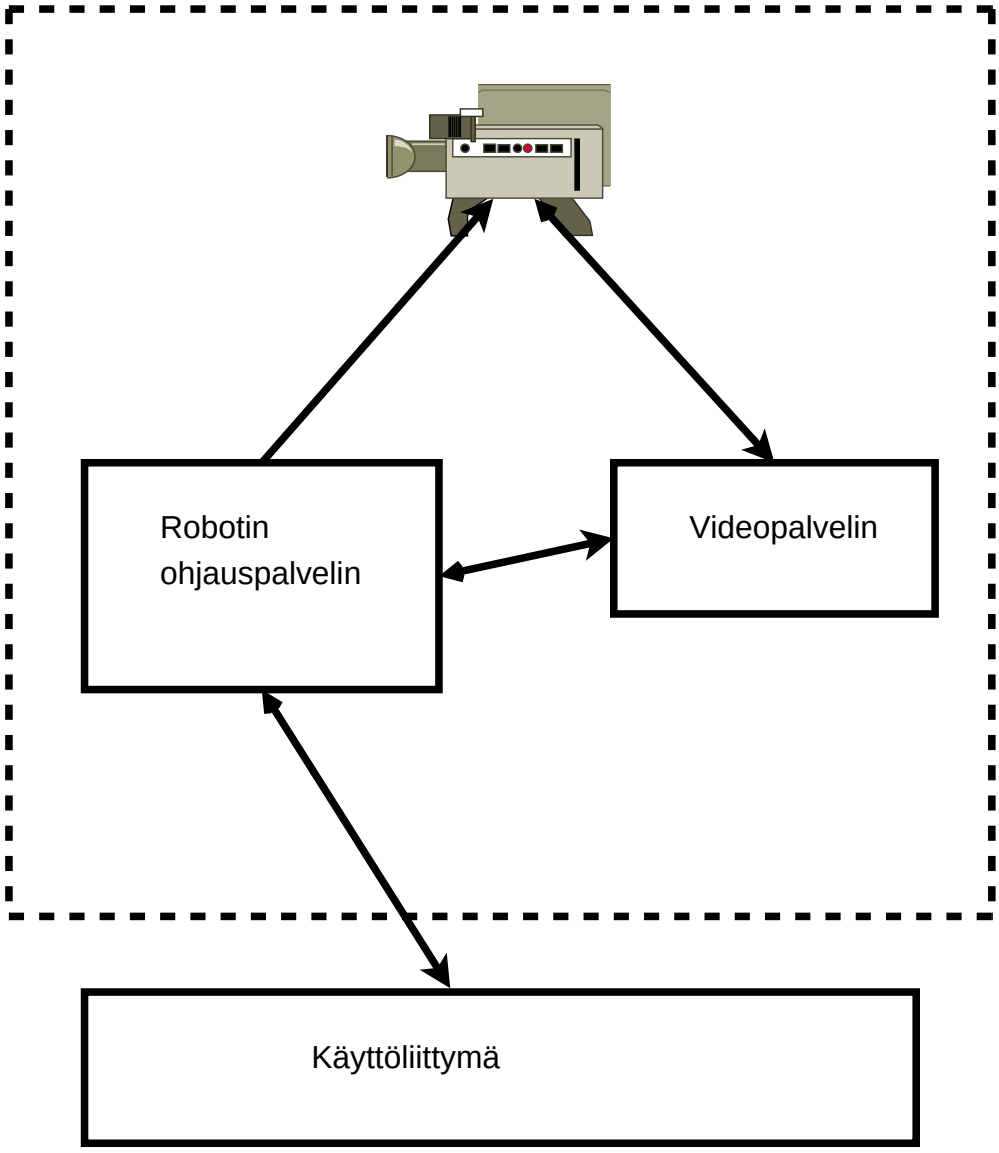
Koska näkymän tapahtumankäsittelysilmukkaa ajetaan omassa säikeessään, tulee näkymän varmistaa, ettei mallin säie päivitä näkymää väärään aikaan. Kirjastotasolla tämä tarkoittaa, ettei GTKMM-kirjaston käyttöliittymäkomponentteja saa päivittää, elleivät kaikki GTKMM-kirjaston komponentit ole päivittyneet. Päivitykset ta-

Kuva 7: Käyttöliittymän luonnos.

pahtuvat tapahtumankäsittelysilmukan kautta, ja uusia tapahtumia saa näin ollen luoda vain kun tapahtumankäsittelysilmukka on valmiina ottamaan uusia tapahtumia vastaan. Näkymän rajapinnassa on mallia varten yksi metodi, jolla voidaan kertoa, minkätyyppinen tieto muuttuu. Näkymän tulee ottaa päivityskutsu vastaan, ja signaloida päivitys itselleen. Päivitys käsitellään vasta, kun tapahtumankäsittelysilmukka on sallitussa kohdassa.

6.3 Palvelin

Robotin mukanaan kuljettamasa sylimikrossa ajetaan kahta palvelunta. Toinen on varsinainen robotin ohjauspalvelin ja toinen on kameran ohjausta varten. Koko järjestelmässä ajettavat prosessit on esitetty kuvassa 8.



Kuva 8: Järjestelmän prosessit.

6.3.1 Videopalvelin

Videopalvelin sisältää kameran videon saamiseen tarvittavan toiminnallisuuden. Robotin ohjauspalvelin liittyy asiakkaaksi videopalvelimen tarjoamaan TCP/IP-palvelinporttiin, ja ottaa videon vastaan. Olemassaolevan videopalvelimen toteutukseen ei tarvitse tässä työssä puuttua. Kameran ohjaus tehdään kuitenkin ohjauspalvelimessa.

Kamera mallinnetaan PTZ-kamerana (pan-tilt-zoom), eli kameraa voidaan kääntää horisontaalisesti (pan), vertikaalisti (tilt) ja sen polttoväliä voidaan muuttaa (zoom). Ohjauspalvelimeen asetetaan hallintaobjekti käsittelemään käyttöliittymästä tulevia PTZ-kameranhallintakomentoja. Hallintaobjektissa on yleisen PTZ-komennon, esimerkiksi kameran kääntämiskomennon, tarvitsema käytännön toteutus. Nämä ovat käytetyn kameran tapauksessa HTTP-pyyntöjä.

6.3.2 Ohjauspalvelin

WLAN-yhteyden tilan tarkkailuun ja puhesyntetisaattorin ohjaukseen luodaan uudelleenkäytettävät luokat. Luokat asennetaan ohjauspalvelimen tapahtumankäsittelysilmukkaan. WLAN-tarkkailu tulee olemaan yhden metodin käsittelijä asiakaskäyttöliittymän pyynnöille. Puhesyntetisaattorin kutsumisen havaittiin olevan estollinen (blocking), joka tarkoittaa ohjelman suorituksen pysähtymistä, kunnes syntetisointi on kokonaisuudessaan suoritettu. Tästä johtuen toiminnallisuus jaetaan käyttäjän syötteet keräävään osaan ja omassa säikeessä ajettavaan puhesyntetisaattorin ohjaukseen.

7 Toteutus

7.1 Rajapinnat

Rajapinnat `IView` ja `IModel` ovat toteutettu abstrakteina luokkina, joissa on vain puhtaan virtuaalisia metodeja ja apuluokkien ja -tietueiden määrityksiä. Tämä on samankaltainen järjestely kuin java-ohjelmointikielen `Interface` määritys. Kyse on moniperinnän hallinnasta, kun halutaan määritellä vain rajapinta. Toiminnallisuus rajapintaluokassa ei ole tarpeellista, eikä suotavaa.

7.2 Mutex-luokkamalli

Mallin tiedon säieturvallista päivitystä ja lukemista varten luotiin luokkamalli `MutexData`. Tämän luokkamallin tarkoituksena on kapseloida mikä tahansa luokka mutexin kanssa siten, että aina kun haluttua tietoa käsitellään, lukitaan siihen liittyävä mutex. Luokkamallin muodostimelle annetaan parametrinä haluttu olio, joka asetetaan luokkamallin sisäiseksi olioksi. Mutex on luokkamallissa niinikään sisäinen olio. Luokkamallissa on vain kaksi metodia, `get_data` ja `unlock`. Ensimmäisellä yritetään lukita mutex ja saada paluuarvona olio, johon pääsy halutaan lukita ajavalle säikeelle. Jälkimmäisellä vapautetaan mutexin lukko. Seuraavassa on yksinkertaistettu luokkamalli:

```
template<class T>
 class MutexData {
  private:
   T data;
   m mutex;
  public:
   T get_data() { m.lock(); return data };
   unlock() { m.unlock() };
}
```

7.3 Malli

7.3.1 Robotilta saatava tieto

Saataessa yhteys robotin ohjauspalvelimeen asennetaan käsittelijät palvelimelta saataville viestipaketeille. Palvelimelta pyydetään lista käytettävistä sensoreita, kameran tiedot ja saatavilla oleva pohjakartta. Lisäksi lähetetään pyynnöt saada jatkuvasti dataa, robotin paikkatiedosta ja sensoreista. Pyynnöt videosta ja reitinsuunnitelusta aktivoidaan vasta näkymän pyytäessä.

Robotin ohjauspalvelimelta saatavat tiedot talletetaan paikallisiin olioihin. Robotin perustieto, kameran tieto, videodata, ja WLAN-informaatio kootaan tietueisiin.

Sensoridata koostuu pisteistä, ja pohjakartta koostuu viivoista, pisteistä ja kartan reunan ilmaisevista viivoista. Näitä kaikkia kuvaa STL (standard template library) vektori-luokkamalli, johon talletetaan jonoksi `Point`-tietueita, joissa on xy-koordinaattipari. Pohjakartan kolme vektoria haetaan erillisillä metodeilla mallilta.

Asetustenhallinnan toteuttaa `ModelConfData`-olio luvussa 6.2.3 kuvatulla tavalla. `ModelConfData`-luokan sisältämät parametrilistat ovat toteutettu STL-luokkamalleilla [35] ja niitä käydään läpi vastaavilla iteraattoreilla. Parametrien arvona tuetaan kokonaislukuja ja boolean-arvoja.

Tietueet, pistevektorit ja asetukset kapseloidaan `MutexData`-luokkamallin oliolla. Kaikki metodit, jotka hakevat näitä tietoja mallilta, palauttavat kapseloidun olion.

7.3.2 Komennot

Pyynnöt videosta ja reitinmäärityksestä muodostavat pyynnön ohjauspalvelimelle, johon saadaan vastauspaketteja halutuin väliajoin. Väliajat ovat muokattavissa asetustenhallinnassa. Muut pyynnöt aiheuttavat vain yhden vastauksen.

ARIA-kirjaston runko-olio `ArClientBase` voi palvelinyhteyden muodostuttua tarkistaa, tukeeko palvelin käsittelyä halutulle pyynnölle. Mallin pyyntömetodit tekevät tarkistuksen, ja muodostavat pyyntöpaketin. Pyyntö kootaan kirjaston metodeilla pakettiolioon `ArNetPacket` tavujonoksi, ja pakettiolio annetaan runko-olion lähetettäväksi.

7.4 Näkymä

Mallin säikeen kutsuessa näkymää tiedon päivittyessä `IView`-rajapinnan metodilla `update`, lukitsee näkymä päivitysmutexin, ja lisää saadun päivityksen tyypin jonoon. Lisäksi signaloidaan GTKMM-kirjaston tapahtumankäsittelysilmukkaa tarkoitukseen luodulla `Dispatcher`-oliolla, jonka jälkeen mutex vapautetaan. Mallin säie jatkaa suoritustaan. Näkymän säie jatkaa suoritustaan, kunnes tapahtumankäsittelysilmukka käsittelee `Dispatcher`-olion signaalin. GTKMM-kirjasto on nyt tilassa, jossa käyttöliittymäkomponentteja voidaan päivittää. Signaalin käsittelyssä käydään läpi jonossa olevat päivitystyypit kutsuen sopivaa näytönpäivitysmetodia tyypin mukaan.

7.4.1 Karttanäyttö

Robotin suunta, sensoridata, pohjakartta ja suunniteltu reitti piirretään OpenGL-kirjaston avulla. Karttanäyttö hakee tiedon mallilta itsenäisesti. Seuraavaksi esitetään vaiheittain karttanäytön piirto:

1. Siirretään koordinaatisto robotin koordinaatteihin ja piirretään kartta.
2. Jokaiselle sensoridatapisteelle lasketaan suhteellinen kohta kartalla robottiin nähden ja piirretään piste.
3. Jokaiselle reitin pisteelle lasketaan suhteellinen kohta kartalla robottiin nähden ja piirretään yhdistetty viiva.
4. Piirretään robottia kuvaava ympyrä ja suuntaosoitin.

Koko piirto suoritetaan metodissa, joka käsittelee myös yleiset ulkoiset (esim. ikkunointijärjestelmältä tulevat) uudelleenpiirtopyynnöt. Pyyntöjä generoidaan sisäisesti näkymässä, kun saadaan päivitysilmoitus mallilta.

Käyttäjä voi asettaa reitinsuunnittelulle kohdepisteen kartalla. Hiirenpainallus kartalla otetaan talteen ja tarkistetaan onko näkymässä reitin asetus päällä. Mikäli on, kutsutaan mallin reitinsuunnittelufunktiota. Reitinmääritys voidaan myös toisella hiirennapilla perua.

7.4.2 Yleinen dialogi

Yleisiä dialogeja käytetään yksinkertaisen syötteen saamiseksi käyttäjältä. Ohjelmasta poistumisen ja törmäyksentarkistuksen poiskytkennän varmistus vaativat syötteenä vain statuksen, hyväksyikö käyttäjä toimenpiteen. Luotu yleisdialogiluokka `GenDialog` tukee ok/cancel/apply-toimintoja, luoden näille painonapin. Toimintoihin kuuluu myös kenttä tekstisyötteille. Luokan muodostimessa voidaan booleanarvoilla ilmoittaa, mitä toimintoja halutaan käyttöön. Luokka periytyy myös apuluokasta `DialogUtil1`, joka tarjoaa funktoripohjan toiminnallisuuden sitomiseen yleisdialogiluokan toimintoihin. Yleisdialogilla ei ole osoitinta näkymäolioon. Yleisdialogin näyttäminen ja syötteen odotus ei estä käyttöliittymän säikeen suoritusta.

7.4.3 Erikoistuneet dialogit

Erikoistuneet dialogit eivät käytä funktoreita, vaan kutsuvat suoraan näkymän julkisia metodeja. Näidenkään dialogien näyttäminen ja syötteen odotus ei pysäytä käyttöliittymän säikeen suoritusta. Erikoistuneilla dialogeille tulee olla osoitin näkymäolioon julkisten metodien kutsumiseksi.

Asetustenhallintadialogi periytyy Glade-kehittimellä luodusta luokasta. Asetusparametrien näyttö pohjautuu GTKMM-kirjaston `TreeView`-luokkaan, joka näyttää asetukset puumaisesti. Puussa on käytännössä kaksi tasoa, osasto ja asetus. Avattaessa näytöllä päätason solmu, joka kuvaa osastoa, saadaan sen alla olevat asetukset. Käyttöä varten täytyy `TreeView`-luokan `TreeModel`-malliluokka alustaa asetustiedolla. Asetushallintadialogi alustetaan osoittimella näkymään. Dialogin ainoa julkinen metodi on `set_confdata`, jolla näkymä asettaa osoittimen mutexilla kapseloituun asetustietoon. Asetukset haetaan kahdessa sisäkkäisessä silmukassa. Ulommassa käydään läpi osastot ja sisemmässä haetaan asetukset. `TreeModel`-olioon luodaan sama hierarkia, joka asetetaan lopuksi `TreeView`-oliolle, tallettaen myös viite asetusolioon. Puu näyttää osastohierarkian, ja siihen liittyvän parametrin arvon, joka on muutettavissa. Kun käyttäjä hyväksyy muutokset, käydään puun malli läpi, päivittäen mallilta saatu asetusolio.

Yhteydenmuodostusdialogi näyttää käyttäjälle tunnistus- ja yhteystiedot ja ok/cancel-napit. Dialogi saa muodostimessa osoittimen malliin ja näkymään. Näkymän osoitinta käytetään vain ohjelmasta poistumiseen, mikäli painetaan cancel-nappia. Ok-napin painalluksella kutsutaan mallin yhteydenmuodostusmetodia dialo-

gin parametreilla, ja käynnistetään mallin säie mikäli yhteydenmuodostus onnistui.

7.5 Palvelin

WLAN-yhteyden tilan tarkkailua ja puhesyntetisaattorin ohjausta varten toteutettiin vastaavasti kaksi luokkaa, `WlanHandler` ja `SpeechHandler`.

WLAN-tarkkailu saadaan palvelimen lähdekoodissa käyttöön antamalla luokkien muodostajalle osoitin ARIA-kirjaston `ArServerBase`-olioon, joka on robotin ohjauspalvelimen runko. Muodostaja rekisteröi ohjauspalvelimen runko-olioon käsittelijän, jota kutsutaan, kun käyttöliittymän asiakasolio pyytää WLAN-tarkkailutietoa. Puhesyntetisaattorin ohjausluokka toimii täysin vastaavasti tässä suhteessa. Molemmissa on myös metodi toiminnallisuuden varmistamista varten. Kyseinen metodi tekee mahdolliset tarvittavat alustukset ja tarkistaa, onko WLAN-tieto tai puhesyntetisaattori saatavilla. Ohjauspalvelin voi tarkistaa metodin paluuarvosta, onko toiminnallisuus todella päällä.

7.5.1 WLAN

WLAN-informaatiota saadaan Linux-kernelin `/proc`-tiedostojärjestelmästä [36], joka tarjoaa suoran kuvan ajonaikaisesta järjestelmästä, esimerkiksi prosesseista ja oheislaitteista. Aukaisemalla tiedosto `/proc/net/wireless` [37] voidaan lukea WLAN-ajurilta tietoa:

Quality - link (linkki) tarkoittaa, kuinka virheettömästä dataa saadaan vastaan, esimerkiksi prosentti oikein vastaanotetuista paketeista.

Quality - level (taso) on signaalin voimakkuustaso.

Quality - noise (kohina) tarkoittaa voimakkuustasoa, jota pienemmällä tasolla ei saada dataa vastaan.

Tieto parsitaan `/proc/net/wireless` -tiedoston ulostulosta säännönmukaisia lausekkeita (regular expression) hyväksikäyttäen. Asiakasolion niin pyytäessä, tieto parsitaan ja pakataan vastaukseksi kolmeksi ARIA-kirjaston kaksitavuiseksi datatyypiksi peräkkäin ja lähetetään.

7.5.2 Puhesyntetisaattori

Puhesyntetisaattorin ohjausluokka `SpeechHandler` ottaa vastaan asiakkaalta tulevat tekstisyötteet ja lisää ne STL-kirjaston `dequeue`-luokkamallilla toteutettuun jonoon. Syntetisaattorin käsittelyn hoitaa `SpeechHandler`issa luotava luokka `SpeechTask`, joka on ARIA-kirjaston säieluokasta periytetty. Luokkaa ajetaan omana säikeenään tarkistaen kahdensadan millisekunnin välein, onko jonossa puhesyntetisaattorille syötettäviä merkkijonoja. `SpeechTask` saa alustuksessa osoittimet syötejonoon ja jonon lukitsemiseen tarvittavaan mutex:iin.

8 Testaus

8.1 Testitapaukset

Ohjelmistolle tehtiin käytännön testejä Lappeenrannan teknillisen yliopiston kuudennessa rakennusvaiheessa. Testauksissa oli mukana kehittäjä Jukka Turusen lisäksi tutkimusapulainen Janne Laaksonen. Testitapauksissa on määritelty testattava ominaisuus ja odotettu lopputulos:

1. Otetaan yhteys käyttöliittymästä palvelimeen väärällä osoitteella. Yhteydenottodialogissa tulee näkyä tieto yhteyden epäonnistumisesta. Yhteydenottodialogi jää esille.

2. Otetaan yhteys käyttöliittymästä palvelimeen oikealla osoitteella. Käyttöliittymän tulee muodostaa yhteys ja piilottaa yhteydenottodialogi.

3. Robotin serverille annetaan käynnistyksen yhteydessä pohjakartta ympäristöstä. Otettaessa käyttöliittymästä yhteys palvelimeen, kartan tulee latautua käyttöliittymän karttanäyttöön.

4. Perusliikkumiskomennot:

   (a) liikkuminen eteen,

   (b) liikkuminen taakse,

   (c) ohjaus vasemmalle ja

   (d) ohjaus oikealle.

   Robotin tulee liikkua ohjauksen mukaisesti. Suoritetaan kahdesti eri nopeusasetuksilla, 50% ja 100%, ja robotin nopeusnäytön tulee näyttää nopeudet suoraan eteenpäin liikkuessa vastaavasti.

5. Kytketään turvallinen ajo käyttöliittymästä pois päältä. Ohjataan robotti kevyttä estettä (esimerkiksi pahvilaatikko) kohti. Robotin tulee osua esteeseen.

6. Kytketään turvallinen ajo käyttöliittymässä päälle. Ohjataan robotti kevyttä estettä kohti. Robotin tulee pysähtyä ennen estettä.

7. Aktivoidaan vaellusmoodi (wander mode). Robotin tulee lähteä liikkumaan autonomisesti. Annetaan robotin kulkea noin 10 sekuntia, ja annetaan pysähtymiskäsky. Robotin tulee pysähtyä.

8. Ohjataan robotti osoittamaan kartan seiniä kohti eri etäisyyksillä. Kaikuluotaimien mittausten osoittimien tulee olla silmämääräisesti arvioiden kartan seinien kohdilla. Toistetaan kolmella erilaisella karttanäytön zoom-asetuksella: perusasetus, ja yksi säätöpykälä ylös- ja alaspäin.

9. Käyttöliittymässä annetaan reitinsuunnittelulle loppupiste. Käyttöliittymän tulee visualisoida robotin kulkema reitti, ja robotin on saavuttava määränpäähänsä.

10. Aktivoidaan videonhaku. Kuvan tulee näkyä videonäytöllä.

11. Ohjataan kameran PTZ-parametrit käyttöliittymän sallimiin ääriasentoihin ja takaisin keskelle:

    (a) Vertikaaliset ääriasennot.

    (b) Horisontaaliset ääriasennot.

    (c) Polttovälin ääriasennot.

    Kameran asennon ja polttovälin tulee käydä ääriasennoissaan ja palata keskelle.

12. Muutetaan asetustenhallinnassa videon päivitysväli 500 millisekuntia 100 millisekunniksi. Kuvan tulee päivittyä nopeammin.

13. Tallennetaan muuttuneet asetukset. Asetustiedostossa tulee videon päivitysvälin muuttua arvoon 100.

14. Puhesyntetisaattorille annetaan käyttöliittymästä käsky syntetisoida puhetta. Robotin ohjauspalvelilmen tulee toistaa syntetisoitu puhe.

15. Viedään WLAN-komponentteja (tukiasema ja mahdollisesti käyttöliittymää ajava sylimikro) niin etäälle toisistaan, että robotin ohjauspalvelimesta saatavassa WLAN-tiedossa tapahtuu oleellisia muutoksia. Jos mahdollista, säädetään WLAN-laitteiden lähetysteho minimiin. Tavoite on saada näytettyä WLAN-informaationa linkistä alle 100% arvoja.

Testitapausten lisäksi mitataan latenssi käyttöliittymältä palvelimeen. Standardi C-kirjaston `gettimeofday`-kutsulla saadaan systeemin aika mikrosekunnin tarkkuudella. Mittaus tehdään ajamalla käyttöliittymää ja palvelinta samalla tietokoneella, jotta eri prosessien saamat systeemin ajat olisivat vertailukelpoisia. Aika talletetaan käyttöliittymässä robotin ohjauskomennon yhteydessä omaan pakettiinsa, joka lähetetään varsinaisen komennon perään. Aikapaketille on palvelimessa oma käsittejänsä, joka purkaa ajan ja laskee erotuksen nykyiseen aikaan.

8.2 Tulokset

Taulukossa 1 on esitetty testien tulokset. Neljä testiä epäonnistui. Turvallista ajoa ei saatu pois päältä, vaikka robotin ohjauspalvelin ilmoittaa konsolissa vastaanottaneensa käskyn. Tästä johtuen myös turvallisen ajon laittoa takaisin päälle ei voitu testata. Horisontaalisessa kameran ohjauksessa havaittiin kameran kääntyvän tietyn pisteen jälkeen vastakkaiselle puolelle. Tämän arvellaan johtuvan tavujen käsittelystä kameran asentoa kuvaavien komentojen pakkauksessa, sillä ongelmana näyttivät olevan arvoalueen -127 - 127 ulkopuolen arvot. Vastaavanlaisia ongelmia on tavattu ARIA-kirjastosta kehityksen aikana. Videon päivityksessä asetus päivittyi asetustiedostoon, mutta videokuva ei päivittynyt nopeammin. Kaikuluotaimien mittausten arveltiin silmämääräisesti olevan kartan seinien kohdalla, vaikka ongelmia oli robotin kohdistamisessa oikein suhteessa karttaan. Kartan kohdistamiseen liittyvät toiminnallisuudet ovat mahdollisia jatkokehityskohteita käyttöliittymään.

Latenssin mittauksessa tulokset vaihtelivat alle millisekunnista noin viiteenkymmeneen millisekuntiin.

Taulukko 1: Testitulokset. Tuloksissa OK tarkoittaa onnistunutta testiä. Tulos voi sisältää lisätarkennusta tai lyhen selosteen testin epäonnistumisesta.

	Testi	**Tulos**
1.	Yhteyden muodostus väärin.	OK
2.	Yhteyden muodostus.	OK
3.	Pohjakartta.	OK
	Perusliikkumiskomennot.	
4.	a) eteen 50%	OK
4.	b) taakse 50%	OK
4.	c) vasen 50%	OK
4.	d) oikea 50%	OK
4.	a) eteen 100%	OK
4.	b) taaakse 100%	OK
4.	c) vasen 100%	OK
4.	d) oikea 100%	OK
4.	50% → 100% nopeuden kasvu	OK
5.	Turvallinen ajo pois päältä.	ei saatu pois päältä
6.	Turvallinen ajo päällä.	ei voitu testata (testi 5.)
7.	Vaellusmoodi.	OK
8.	Kaikuluotaus.	OK, kohdistus hankala
9.	Reitinsuunnittelu.	OK
10.	Video.	OK
	Kameran ohjaus.	
11.	a) Vertikaaliset asennot.	OK
11.	b) Horisontaaliset asennot.	ääriasento väärällä puolella.
11.	c) Polttoväli.	OK
12.	Videon päivitys.	ei päivittynyt nopeammin.
13.	Asetuksen tallennus.	OK
14.	Puhesyntetisaattori.	OK
15.	WLAN-informaatio.	OK

9 Pohdinta

Pääasiallisia perusteita ohjaimen poisjäännille MVC-mallista on kaksi: toinen käytetyistä työkaluista johtuva ja toinen on suunnittelullinen päätös. Ensisijainen syy on käytetyn GTKMM-kirjaston ja Glade-kehittimen tuottaman koodin luonne. Käyttäjän syötteet saadaan vastaan käyttöliittymäolioiden metodeilla. Esimerkiksi nappia painettaessa aktivoidaan metodi kyseisestä nappi-oliosta, joka myös hoitaa visualisoinnin. Näin ollen syötteet ovat hankalasti erotettavissa omiksi olioikseen. Voidaan ajatella, että ohjain ja näkymä ovat jo samaa oliota. Kuten vertailussa [25] todetaan, käyttöliittymäkomponenttikirjastoja käytettäessä erotus on hankalampaa, ja näin on tässäkin tapauksessa. Toissijainen syy on arkkitehtuurin pitäminen yksinkertaisena. Vaadittava toiminnallisuus arvioitiin siinä määrin yksinkertaiseksi, että arkkitehtuuria kannattanee pitää kevyenä laajennettavuuden kärsimättä.

Kun robottiin tulee enemmän toiminnallisuutta (esim. lisäkameroita tai tarttumakäsiä), ja saatavan informaation määrä alkaa kasvaa, on oletettavissa että visualisointi ja sen ylläpito hankaloituu oleellisesti [1] [8]. Näissä tapauksissa PAC-arkkitehtuuri voi ylläpidollisesti soveltua paremmin, koska käyttöliittymää jaetaan pienempiin kokonaisuuksiin. Lisäksi GTKMM-kirjasto soveltuu paremmin presentation-komponentin toteuttamiseen [25]. Lisäkustannuksena on koko toteutuksen kompleksisuus.

Menetelmä, jolla malli ilmoittaa tiedon päivittymisestä näkymälle, tulisi muuttaa käyttämään tarkkailija-oliota (observer) [33]. Tämä sallisi toisistaan riippumattomien näkymä-olioiden vastaanottaa tieto päivittymisestä. Useamman näkymän tapauksessa täytyy varmistaa mallin pyyntömetodien säieturvallisuus. Tällä hetkellä oletetaan, että vain yksi näkymäsäie tekee pyyntöjä.

Glade:n luoma XML-kuvaus käyttöliittymästä on myös mahdollista ottaa käyttöön ajonaikana `libglademm` -kirjastoa käyttäen. Näkymän ohjelmointia tämä muuttaa siten, että viittaus tiettyyn graafiseen komponenttiin on haettava kirjaston metodeilla käyttöliittymäkuvauksen parsimisen jälkeen, kun nyt ne saadaan suoraan luokan sisäisinä muuttujina. Tämä mahdollistaa GUI:n ulkonäön muuttamisen Glade:lla ilman koko ohjelman uudelleen kääntämistä.

`MutexData` -luokkamalli sallii kapseloitavaksi annetun datan muuttamisen, joka ei useimmissa tapauksissa ole toivottavaa, sillä näyttö haluaa ainoastaan lukea sellaista mallin dataa, jota tarvitsee synkronoida tällä tavoin.

Ohjausserveriin kytkeytyvät oliot voidaan kirjastoida. Kirjaston perustan voivat luo-

da `WlanHandler` ja `SpeechHandler` -luokat. Kirjastoa voitaisiin jatkossa laajentaa halutuilla toiminnallisuuksilla.

Asetusten hallinta on rajoittunut yksinkertaisiin tietotyyppeihin. Hallinta ei tarjoa mahdollisuutta hallita tilannetta, jossa yhden asetuksen muutos vaikuttaa toisiin parametreihin. Tämä olisi toteutettavissa tallentamalla hallintaan asetuksen määrittelyn yhteydessä haluttu käsittelijä, jota kutsutaan kutsutaan arvon muuttuessa. Toiminnallisuutta osastojen hallintaan olisi ollut valmiinakin ARIA-kirjaston asetustenkäsittelyssä, joten oman toteutuksen tekeminen rinnalle jää kyseenalaiseksi.

Käyttöliittymän ja palvelimen välinen 50 millisekunnin latenssi on selvästi käyttökelpoisissa rajoissa, jos oletetaan tavoitteeksi pysyä alle kuudensadan millisekunnin, josta alkaen voidaan havaita käyttäjän sopeutumista viiveeseen [12]. Vaikka arvoon laskettaisiin lisäksi tyypillinen huonoimman tapauksen latenssi 70 millisekuntia, pysyy vasteaika alle kahdessasadassa millisekunnissa [20]. Testausmenetelmä antanee huonoimman tapauksen viiveen. Prosesseja ajettiin rinnan samalla yksiprosessorisella tietokoneella, joten prosessien saama suoritusaika vaikuttaa, miten nopeasti palvelin saa syötteen käsiteltyä. Lisäksi mittaukselle on oma pakettinsa, joka lisää viivettä.

10 Johtopäätökset

Työssä onnistuttiin toteuttamaan käyttäjälle näkyvä vaadittu MobileEyes-sovelluksen toiminnallisuus. Lisäksi onnistuttiin WLAN-yhteyden tilan ja puhesyntetisaattorin ohjauksen toteutus uusina ominaisuuksina käyttöliittymään ja palvelimeen.

ARIA-kirjaston lähdekoodi on GPL:n alainen. Tämä tarkoittaa, että ARIA-kirjastoa käyttävän ohjelman lähdekoodin on oltava julkisesti saatavilla, mikäli ohjelmistoa jaetaan eteenpäin. GTK+- ja GTKMM-kirjastot ovat LGPL:n (Library GPL) alaisia. Kyseinen lisenssi sallisi lähdekoodin pitämisen suljettuna edellyttäen, että ohjelma on dynaamisesti linkikettävissä kyseiseen kirjastoon [38]. ARIA-kirjaston GPL kuitenkin on tässä tapauksessa hallitsevampi.

Mallin ja näkymän erotus toisistaan selkeyttää rakennetta. Lisäksi keskinäiset riippuvuudet vähenevät ja jatkokehitystä voidaan tehdä komponenttikohtaisesti. Itse mallin sisäinen toteutus ei ole selkeän modulaarinen ja uusien dialogien tai muiden näyttöjen hallinta voi olla sekavaa. Vaatimuksena ollut helppo laajennettavuus ei näin ollen täysin toteutunut.

Jatkokehityskohteita on kaksi:

- malli-näkymä-rajapinnan muuttaminen käyttämään tarkkailija-oliota ja
- kartan kohdistamisen mahdollistava toiminnallisuus käyttöliittymässä.

Ensimmäinen kohde lisää ohjelmiston modulaarisuutta ja toinen kohde parantaa oleellisesti käytettävyyttä.

Lähteet

[1] Fong, Terrence & Thorpe, Charles. 2001. Vehicle teleoperation interfaces. *Autonomous Robots*, 11(1):9–18. ISSN 0929-5593.

[2] Habib, Maki K. 2000. Teleoperation and collaborative control. *Sixth International conference on Intelligent and Autonomous systems (IAS-6)*, Venice, Italy.

[3] Scholtz, J. 2003. Theory and evaluation of human robot interactions. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, 125. ISBN 0-7695-1874-5.

[4] Hainsworth, D. W. 2001. Teleoperation user interfaces for mining robotics. *Autonomous Robots*, 11(1):19–28. ISSN 0929-5593.

[5] Marques, L., Rachkov, M. & Almeida, de A.T. 2002. Mobile pneumatic robot for demining. *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA '02.*, 4:3508–3513.

[6] Dario, P & Menciassi, A. 2002. Robotics for surgery. *Proceedings of the Second Joint EMBS/BMES Conference*, 3:1813–1814. ISBN 0-7803-7612-9. ISSN 1094-687X.

[7] Meng, Cai, Wang, Tianmiao, Chou, Wusheng, Luan, Sheng, Zhang, Yuru & Tian, Zengmin. 2004. Remote surgery case: robot-assisted teleneurosurgery. *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, 1:819–823. ISSN 1050-4729.

[8] Kadous, M. Waleed, Sheh, Raymond Ka-Man & Sammut, Claude. 2006. Effective user interface design for rescue robotics. *Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 250–257. ISBN 1-59593-294-1.

[9] Ellis, R. E., Ismaeil, O. M. & Lipsett, M. G. 1996. Design and evaluation of a high-performance haptic interface. *Robotica*, 14:321–327.

[10] Milgram, P., Zhai, S., Drascic, D. & Grodski, J. 1993. Applications of augmented reality for human-robot communication. *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems '93*, 3:1467–1472. ISBN 0-7803-0823-9.

[11] Olivares, R., Zhou, C., Adams, J.A. & Bodenheimer, R. 2003. Interface evaluation for mobile robot teleoperation. *Proceedings of the ACM Southeast Conference (ACMSE03)*, 112–118.

[12] Ando, N., Lee, Joo-Ho & Hashimoto, H. 1999. A study on influence of time delay in teleoperation. *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 317–322, New York, NY, USA. ACM Press. ISBN 0-7803-5038-3.

[13] Borenstein, J. & L., Feng. 1995. Umbmark: A benchmark test for measuring odometry errors in mobile robots. *Proceedings of the 1995 SPIE Conference on Mobile Robots*, 2807–2812. ISBN 0-7803-4300-X.

[14] Siegwart, Roland & Nourbakhsh, Illah R. 2004. *Introduction to Autonomous Mobile Robots*. MIT Press. ISBN 0-262-19502-X.

[15] Bruce, J. & Veloso, M. 2002. Real-time randomized path planning for robot navigation. *IEEE/RSJ International Conference on Intelligent Robots and System*, 3:2383– 2388. ISBN 0-7803-7398-7.

[16] Krishnaswamy, Gita & Stentz, Anthony (Tony). Resolution independent grid-based path planning. Technical Report CMU-RI-TR-95-08, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 1995.

[17] Fox, D., Burgard, W. & Thrun, S. 1997. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4:23–33. ISSN 1070-9932.

[18] Winfield, Alan F.T. 1999. Future directions in tele-operated robotics. *Telerobotic Applications*, 147–162. ISBN 1-86058-235-4.

[19] Winfield, A.F.T & Holland, O.E. 2000. The application of wireless local area network technology to the control of mobile robots. *Microprocessors and Microsystems*, 23(10):597–607. ISSN 0141-9331.

[20] Del Puerto Riquelme, C., Garcia, R & Malumbres, M.P. 2004. Analyzing the behavior of a real-time telerobotic system on ieee 802.11b wireless networks. *2004 IEEE International Conference on Industrial Technology*, 963–968. ISBN 0-7803-8662-0.

[21] Munir, S. & Book, W.J. 2002. Internet-based teleoperation using wave variables with prediction. *IEEE/ASME Transactions on Mechatronics*, 7:124–133. ISSN 1083-4435.

[22] Reenskaug, Trygve. 1979. Thing-model-view-editor an example from a planningsystem.

[23] Krasner, Glenn E. & Pope, Stephen T. 1988. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *Journal of Object-Oriented Programming*, 1:26–49. ISSN 0896-8438.

[24] Coutaz, Joëlle. 1987. Pac, an object-oriented model for dialog design. *INTERACT 87 - 2nd IFIP International Conference on Human-Computer Interaction.*, 431–436.

[25] Hussey, A. & Carrington, D. 1997. Comparing the mvc and pac architectures: a formal perspective. *IEE Proceedings-Software Engineering.*, 144(4):224–236. ISSN 1364-5080.

[26] Debian [projektin kotisivu], 2006 [viitattu 12.10.2006]. Saatavissa: `http://www.debian.org/`.

[27] Gnu general public license, 2006 [viitattu 12.11.2006]. Saatavissa: `http://www.gnu.org/licenses/gpl.html`.

[28] Gtkmm [projektin kotisivu], 2006 [viitattu 21.10.2006]. Saatavissa: `http://www.gtk.org/`.

[29] Gtk+ [projektin kotisivu], 2006 [viitattu 12.10.2006]. Saatavissa: `http://www.gtk.org/`.

[30] Glade [projektin kotisivu], 2006 [viitattu 12.10.2006]. Saatavissa: `http://glade.gnome.org/`.

[31] Cvs, 2006 [viitattu 12.10.2006]. Saatavissa: `http://www.nongnu.org/cvs/`.

[32] Doxygen [projektin kotisivu], 2006 [viitattu 12.10.2006]. Saatavissa: `http://www.stack.nl/ dimitri/doxygen/`.

[33] Gamma, Erich, Helm, Richard, Johnson, Ralph & Vlissides, John. 1995. *Design patterns Elements of Reusable Object-Oriented Software*. Addison-Wesley. ISBN 0201633612.

[34] Opengl [organisaation kotisivu], 2006 [viitattu 18.10.2006]. Saatavissa: http://www.opengl.org/.

[35] Standard template library programmer's guide, 2006 [viitattu 22.10.2006]. Saatavissa: http://www.sgi.com/tech/stl/index.html.

[36] The linux /proc filesystem as a programmers' tool, 2005 [viitattu 18.10.2006]. Saatavissa: http://www.linuxjournal.com/article/8381/.

[37] Wireless extensions for linux, 1997 [viitattu 18.10.2006]. Saatavissa: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html.

[38] Gnu lesser general public license, 2006 [viitattu 12.11.2006]. Saatavissa: http://www.gnu.org/copyleft/lesser.html.