TAMPEREEN
AMMATTIKORKEAKOULU

AIRBORNE LIDAR

AUTONOMISESSA UAV:SSA

Jesse Tikka

Opinnäytetyö
Toukokuu 2016
Tietotekniikan koulutusohjelma
Tietoliikennetekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Tietoliikennetekniikka

JESSE TIKKA:
Airborne LIDAR autonomisessa UAV:ssa

Opinnäytetyö 36 sivua, joista liitteitä 8 sivua
Toukokuu 2016

Opinnäytetyön aiheena oli suunnitella LiDAR-yksikköä käyttävä autonominen nelikopteri, joka pystyy toimimaan luolastoissa sekä muissa sisätiloissa. Nykyisellään tällaiset ratkaisut ovat mittatilaustyönä tehtäviä, kalliita kertaostoksia.

Sekä LiDAR- että nelikopteritekniikka ovat alati kehittyvässä tilassa. Tämä on otettu huomioon ja nelikopteria suunniteltaessa päädyttiin ROS-järjestelmään. Täten järjestelmä on tulevaisuudessa laajennettavissa erikseen ohjelmoitavilla ROS-moduuleilla.

Työ päätettiin toteuttaa simuloimalla nelikopterin käyttäytymistä 3D-mallinnetussa ympäristössä. Suurimmaksi haasteeksi työn kannalta osoittautui Linux Ubuntu-käyttöjärjestelmän ja ROS-moduulien yhteensovittaminen.

Lopputuloksena syntyneessä nelikopterisimulaatiossa nelikopteria on mahdollista ohjata MAVLink-moduulia käyttäen, joka mahdollistaa autonomiset sovellukset. Simulaattoria on myös mahdollista käyttää erilaisten autopilottien sekä antureiden kanssa.

Asiasanat: LiDAR, airborne, UAV, autonominen

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT engineering
Telecommunications and networks

JESSE TIKKA:
Airborne LIDAR in an autonomous UAV

Bachelor's thesis 36 pages, appendices 8 pages
May 2016

The subject of the thesis was to design an autonomous quadcopter which uses LiDAR-unit for simultaneous localization and mapping while mapping caves and other inside structures. At the time, such solutions are custom-designed, expensive one-time purchases.

Both LiDAR- and the quadcopter technology are in constantly evolving state. This has been taken into account when designing the quadcopter opting ROS system. Thus, the system is expandable in the future with individually programmable ROS modules.

It was decided to carry out the thesis by simulating the behavior of the quadcopter in 3D-modeled environment. The biggest challenge turned out to be the work of the coordination of the Linux Ubuntu operating system and the ROS modules.

In the resulting quadcopter simulation, the quadcopter can be operated with MAVLink module, which makes various autonomic applications possible. Simulator can also be run with support for various types of sensors and autopilot-units.

Key words: LiDAR, airborne, UAV, autonomic

SISÄLLYS

1 JOHDANTO 6
2 LiDAR-TEKNIIKKA 7
2.1 Airborne LiDAR 7
3 NYKYTILANNE 9
3.1 Markkinoilla olevat LiDAR-yksiköt 9
3.1.1 Hokuyo-laseretäisyysanturit 9
3.1.2 Velodyne-laseretäisyysanturit 10
3.2 Kaivosteknologia 11
3.2.1 CSIRO-lasermittaukset 11
3.2.2 GeoSLAM–sisätilalaseretäisyysmittarit 13
3.3 Autonomiset UAV:t 15
3.3.1 Autonomisesti lentävä siipilennokki 15
3.3.2 Objekteja välttävä nelikopteri 16
4 LAITTEISTO 17
4.1 TeraRanger-anturit 17
4.1.1 TeraRanger One-anturi 17
4.1.2 TeraRanger Tower-järjestelmä 18
4.2 Lentokontrolleri 19
4.2.1 Navio2-lentokontrolleri 19
4.3 Prosessointi 20
5 OHJELMOINTI 21
5.1 ROS-käyttöjärjestelmä 21
6 TESTAUS 23
7 POHDINTA 26
LÄHTEET 27
LIITTEET 28
Liite 1. Simulaattorin käyttöönotto 28
Liite 2. Merkkiseuraaja 30

LYHENTEET JA TERMIT

2D	Kaksiulotteinen
3D	Kolmiulotteinen
LiDAR	Light Detection and Ranging
UAV	Unmanned Aerial Vehicle
IMU	Inertia Measurement Unit
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
SLAM	Simultaneous Localization and Mapping
UART	Universal Asynchronous Receiver/Transmitter
I2C	Inter-Integrated Circuit
ADC	Analog to Digital Converter
PWM	Pulse Width Modulation
ROS	Robotic Operating System
SITL	Software in the Loop
MAV	Micro Air Vehicle
UDP	User Datagram Protocol
Autonominen	Itsestäänohjautuva

1 JOHDANTO

LiDAR-tekniikka on laajentumassa kuluttajaystävällisiin, verrattaen halpoihin tuotteisiin. Näiden, helpommin saatavilla olevien, turvallisempien ja kevyempien tuotteiden myötä voidaan kartoitustekniikkaa kehittää uusiin suuntiin.

Tässä opinnäytetyössä tutkitaan autonomista lennokkitekniikkaa ja kuinka verrattaen halvan 2D-LiDAR-järjestelmän voi liittää autonomisesti lentävään alukseen.

LiDAR-ratkaisun on tarkoitus toimia GPS:n tukena. Navigoitaessa tilassa, johon GPS-signaali ei kanna, tarvitaan tieto minne voi turvallisesti lentää. Ennen tähän on käytetty ääniluotausantureita, mutta laseretäisyysanturit ovat huomattavasti tarkempia kuin muut tunnetut etäisyysmittaustavat.

Koska ratkaisua tullaan samanaikaisesti hyödyntämään muunmuassa luolastojen ja rakennusten 3D-mittaamisessa, täytyy lopputuotteen olla mahdollisimman pieni ja kevyt.

2 LiDAR-TEKNIIKKA

LiDAR-tekniikka käyttää laseria mittaamaan etäisyyttä kohteeseen. Se toimii kuten kaikuluotain tai tutka, mutta äänen tai radioaaltojen sijaan LiDAR-yksikkö käyttää valoa. Aallonpituudet 532nm ja 1064nm ovat LiDAR-tekniikassa yleisesti käytettyjä, koska ne sijoittuvat spektrin vihreälle ja infrapuna-asteikolle. Nämä aallonpituudet heijastuvat parhaiten takaisin kasvillisuudesta. LiDAR-dataa voidaan kerätä maasta kolmijalalla, lennosta lentokoneella tai dronella sekä avaruudesta satelliittien välityksellä.

2.1 Airborne LiDAR

Useimmat Airborne LiDAR-järjestelmät pitävät sisällään LiDAR-anturin, GPS-vastaanottimen, IMU-yksikön (eng. intertia measument unit), tietokoneen ja tallennusratkaisun.

LiDAR-järjestelmä projisoi lasersäteen pulssin peiliin, joka ohjaa säteen lentävän alustan (yleensä siipilennokki tai helikopteri) alapuolelle. Säde skannataan reunasta reunaan samalla kun lennokki lentää yli kartoitettavan kohteen, mitaten 20000-150000 pistettä sekunnissa. Lasersäteen osuessa kohteeseen, heijastuu se takaisin peiliin. Lennokista lähtevän ja lennokkiin palaavan pulssin aikaerotus mitataan. Kun LiDAR-tulos on saatu, kerätty data jälkikäsitellään ja LiDAR-aikaviivemittaukset lähtevän ja palaavan pulssin suhteesta muutetaan matkaksi, joka korjataan lennokin GPS-vastaanottimeen, IMU-yksikköön sekä mahdollisiin maassa sijaitseviin GPS-yksiköihin.

GPS-järjestelmä määrittää lennokin sijainnin pituus-, leveys- ja korkeussuunnassa, jotka tunnetaan myös nimillä x-, y- ja z-koordinaatit. LiDAR-anturi kerää valtavan määrän dataa ja yksi kokonainen kartoitus voi helposti tuottaa miljardeja pisteitä, vastaten useita teratavuja tietoa.

IMU-yksikköä käytetään mittaamaan lennokin asentoa avaruudessa. Nämä mittaukset tallennetaan asteina äärimmäisen tarkasti kaikista kolmesta ulottuvuudesta kallistumisena (eng. roll), nyökkäämisenä (eng. pitch) ja kääntymisenä (eng. yaw).

LiDAR-yksikön mittaamaa dataa voidaan edelleen kehittää käyttämällä vaihtoehtoisia jälkikäsittelymetodeja, joista osa voidaan automatisoida, osan pysyessä manuaalisina.

Edelleenprosessonti käyttää useaa paluusignaalia jokaisesta lähtevästä pulssista. Arvioimalla usean paluupulssin aikaeroja, jälkikäsittelyjärjestelmä pystyy erottelemaan rakennukset ja rakennelmat, kasvuston ja maan muodot. Tätä prosessia käytetään poistamaan pinnan ominaisuuksia kuten puita tai kasvillisuutta ja tuottamaan raakoja maanpinnan muotoja.

3 NYKYTILANNE

Useimmat ammattikäyttöön tehdyt UAV:t on tehty keräämään tietoja maan muodoista avokaivoksissa ja –louhoksissa. Jotta kokonaiskuvan luolastoissa autonomisesti toimivien dronei'n nykytilanteesta voi muodostaa, täytyy tarkastella kolmea oleellisesti tärkeää osa-aluetta.

3.1 Markkinoilla olevat LiDAR-yksiköt

Valtaosa markkinoilla olevista LiDAR-yksiköistä eivät kokonsa puolesta sovellu käytettäväksi nelikopterin kanssa. Jos raha ei ole este, löytyy markkinoilta kuitenkin joitakin LiDAR-yksiköitä, jotka ovat sopivat lennokin kanssa käytettäväksi.

3.1.1 Hokuyo-laseretäisyysanturit

Hokuyo on Japanilainen anturitekniikkaa valmistava yritys. Yhtiön UTM-30LX LiDAR-yksikkö on vakaassa suosiossa autonomisten robottien maailmanlaajuisessa kehityksessä. Yksikön 30 metrin toimintasäde toimii 270° skannauskulmalla. Yksikkö painaa 370 grammaa ja pystyy skannaamaan 43200 pistettä sekunnissa. (Hokuyo. UTM-30LX.)

KUVA 1. Hokuyo UTM-30LX-anturi (Hokuyo. UTM-30LX.)

3.1.2 Velodyne-laseretäisyysanturit

Velodyne valmistaa tarkkoja ammattiluokan LiDAR-yksiköitä. Heidän tunnetuin tuotteensa on HDL-64E, jota käytetään muunmuassa Google-auton autonomiseen navigointiin. Vaikka laite pystyykin kuvaamaan ympäristöään erittäin tarkasti, on sen käyttäminen autonomissa UAV-tarkoituksessa hankalaa laitteen painon vuoksi.

KUVA 2. HDL-64E-laseretäisyysmittari (Velodyne. HDL-64E.)

Yhtiö on kuitenkin kehittänyt kiekkomaisen Puck Lite LiDAR-anturin. Kyseinen yksikkö painaa alle 500 grammaa, ja on erittäin käyttökelpoinen SLAM UAV-tarkoituksessa. Tuote on kuitenkin nykyisellä 8000 euron hintalapulla varustettuna varsin kallis.

KUVA 3. Puck Lite-laseretäisyysmittari (Velodyne. Puck LITE.)

3.2 Kaivosteknologia

Useimmissa tapauksissa kaivosten 3D-mallintaminen tehdään käsin kuljetettavalla laitteella tai robotilla, johon on sisällytetty kaivosta kuvaava yksikkö.

3.2.1 CSIRO-lasermittaukset

Commonwealth Scientific and Industrial Research Organisation (CSIRO) on 3D-mallintanut 17 km kultakaivostunneleita Australiassa vuonna 2011. Tunnelit skannattiin erikseen tehtävään tarkoitetulla robotilla, johon oli asennettu kolme LiDAR-yksikköä. Vaakatasossa toimivan yksikön tehtävä oli pitää robotti keskellä kaivostunnellia. Kahden pystytasossa olevan yksikön tehtäväksi jäi skannata tunneli. Kyseinen tapa jättää robotin kulkusuunnassa keskelle kaistan, joka ei mahdu kuvaan. Tämä johtuu pystysuunnassa kuvaavien LiDAR-yksiköiden näkökentästä. Tähän kaivosrobottiin LiDAR-yksiköt valmisti SICK, joka tunnetaan raskaan sarjan teollisuusautomaatiolaitteistaan.

KUVA 4. CSIRO-kaivosrobotti (Youtube. CSIRO – Mapping the Northparkes Mine.)

CSIRO on myös tehnyt prototyypin UAV-yksiköstä, jolla on tarkoitus kuvata voimansiirtolinjoja sekä rakennelmia. Prototyypin nimi on Hovermap ja sen on mahdollista kuvata 41600 pistettä sekunnissa 30 metrin säteellä. Prototyypin painoksi on ilmoitettu 1130 grammaa, mutta toistaiseksi julkaisupäivämäärää ei ole ilmoitettu. (CSIRO. Hovermap – Autonomous Systems.)

KUVA 5. Hovermap-yksikkö (CSIRO. Hovermap – Autonomous Systems.)

3.2.2 GeoSLAM–sisätilalaseretäisyysmittarit

GeoSLAM on eurooppalainen yhtiö, joka valmistaa maanmittaukseen liittyviä tuotteita. Vuonna 2014 GeoSLAM julkaisi käsinkuljetettavan LiDAR-yksikön: Zeb1, jota heiluttamalla saadaan yksikkö mallintamaan ympäristö. Tuotteessa LiDAR-yksikkö on kiinnitetty jousen avulla sauvan päähän ja tätä edestakaisin heiluttamalla luodaan 3D-kuva ympäristöstä. Tuotetta käytetään sisätilojen ja kaivostunneleiden kuvaamiseen ja mallintamiseen.

KUVA 6. Zeb1 LiDAR-yksikkö (GeoSLAM. Zeb1.)

Vuonna 2015 GeoSLAM julkaisi Zeb-revo-yksikön, joka on kuin Zeb1, mutta yksikössä on pyörivä sensori. Tarkoituksena on helpottaa kuvaamista siten, ettei yksikköä tarvitse enää heiluttaa sauvan päässä.

KUVA 7. Zeb-Revo LiDAR-yksikkö (GeoSLAM Zeb-Revo.)

Molemmat GeoSLAMin tuotteet ovat IP64-standardin mukaan suojattuja pölyltä ja roiskeilta. Tuotteet käyttävät pohjanaan Hokuyo UTM-30LX 2D-LiDAR-anturia. (GeoSLAM Zeb-Revo.)

3.3 Autonomiset UAV:t

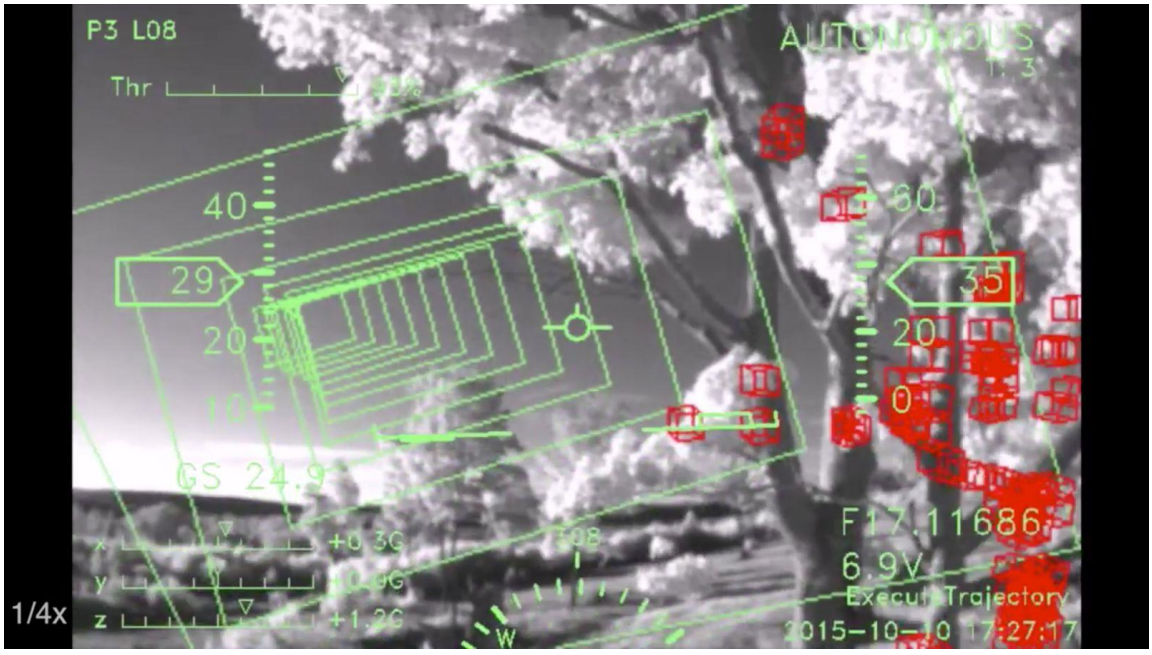
Koska täysin autonomista kaupallista UAV:ta ei ole vielä markkinoilla, täytyy tutustua UAV:ssa käytettävään autonomiikkaan esimerkkitapausten pohjalta.

3.3.1 Autonomisesti lentävä siipilennokki

Tohtorikanditaatti Andrew Barry ja Professori Russ Tedrake ovat työskennelleet algoritmin parissa, joka saa siipimallisen lennokin lentämään autonomisesti.

Järjestelmä käyttää kahta stereoskooppista kameraa mittaamaan objekteja, jotka sijaitsevat tasan 10 metrin päässä kamerasta. Tällä tavoin pystytään karsimaan prosessoitavan tiedon määrää niin, että se on mahdollista käsitellä reaaliajassa. Kameroilta saatava tieto syötetään kahteen neliytimisellä prosessorilla varustettuun tietokoneeseen, jossa tieto käsitellään sitä varten luodulla kuvantunnistusalgoritmilla. (UAV Experts. This Dramatic High-Speed Drone Chase Is Actually A Demo Of MIT's Self-Flying UAV.)

Kyseisellä algortimilla varustettu järjestelmä on saatu onnistuneesti lentämään marraskuussa 2015.



KUVA 8. Näkymä stereoskooppisen kameran silmin (Youtube. Drone Autonomously Avoiding Obstacles at 30 MPH.)

3.3.2 Objekteja välttävä nelikopteri

Maisterikanditaatit Naor Yadai ja Yozef Zohar ovat avoimessa Ariel yliopistossa Israelissa luoneet objekteja välttelevän ja siten myös autonomisen UAV:n. (Youtube. Autonomous UAV Project.)

Ratkaisu käyttää neljää infrapunakameraa, joilta saatava data syötetään Teensy 3.1-alustalle, jossa se käsitellään. Mikäli objekti on liian lähellä UAV:tä, kykenee se itse tekemään tarvittavan korjauksen lentosuuntaan.

KUVA 9. Nelikopterin lentotesti (Youtube. Autonomous UAV Project.)

4 LAITTEISTO

Nelikopterin perusosien lisäksi tarvitaan LiDAR-anturi, lentokontrolleri sekä tietokone, joka pystyy ohjaamaan nelikopteria sekä siihen liitettyjä antureita.

4.1 TeraRanger-anturit

Teraranger-tuotteet on kehitetty yhteistyössä CERN:n kanssa. Yhtiön kaikki tuotteet käyttävät silmäystävällistä LED-tekniikkaa, jossa operoidaan erittäin pienillä tehoilla. Tämä mahdollistaa ihmisten ja muiden elävien organismien toiminnan mittaustilassa turvallisesti ilman erikseen vaadittavaa silmäsuojausta. (Teraranger. Technology advantages.)

Teraranger-sensorit toimivat korkeammilla taajuuksilla kuin normaalit LiDAR-sensorit. Tämä mahdollistaa usean sensorin peräkkäisen liipaisun, jolla vältetään signaalien keskenään tuottama häiriö.

4.1.1 TeraRanger One-anturi

TeraRanger One-anturi on kevyt, korkealuokkainen etäisyysmittausanturi, joka pohjautuu infrapunaa käyttävään Time-of-Flight (ToF) –teknologiaan. Anturi pystyy mittaamaan 14 metrin päähän, ±0,5 cm tarkuuteen precision-tilassa. Precision-tilassa päivitysnopeus on 600 Hz. Anturi painaa 8-10 grammaa riippuen käytettävästä koteloinnista. (Teraranger. TeraRanger One.)

KUVA 10. TeraRanger One-anturi (Teraranger. TeraRanger One.)

4.1.2 TeraRanger Tower-järjestelmä

TeraRanger Tower on kahdeksan TeraRanger One –anturia liitettyinä TeraRanger Hub-keskittimeen. Näin pystytään luomaan 360° astetta kattava LiDAR-luotaus ilman liikkuvia osia ja päästään täten korkeampaan näytteenottotaajuuteen kuin pyörivällä LiDAR-yksiköllä. Järjestelmä vastaa täten nopeammin saatuun tietoon, näytteenottotaajuuden ollessa 270 Hz.

Järjestelmä painaa 130 grammaa, joka tekee siitä erinomaisen nelikopterikäyttöön. Koska järjestelmä on verrattaen kevyt, nelikopteri tarvitsee vähemmän nostovoimaa, joka tarkoittaa että moottoreita voidaan pyörittää hitaammin. Tämän ansiosta sekä akkukesto että lentoaika pitenevät. Järjestelmän hinta on 1140 euroa.

KUVA 11. TeraRanger Tower 8-kameran yksikkö (Teraranger. TeraRanger Tower.)

4.2 Lentokontrolleri

Lentokontrolleri on pieni piirilevy, jolla on vaihtelevia ominaisuuksia. Sen tehtävä on säädellä moottoreiden kierroslukua sisääntulon perusteella. Ohjaajan käsky liikuttaa UAV:tä eteenpäin syötetään lentokontrolleriin, joka vastaavasti määrittää, kuinka moottoreita tulee ohjata.

Suurin osa lentokotrollereista käyttää erilaisia antureita täydentämään laskujaan. Näitä antureita ovat kaikki yksinkertaisesta suunnanmuodostukseen käytettävästä gyroskoopista, automaattiseen lentokorkeuden hakuun käytettävään ilmanpainemittariin. GPS-signaalia voidaan käyttää paikantamisen lisäksi myös autopilot-tarkoituksessa. (Tom’s Hardware. Multi-Rotors And The Hardware You Need.)

4.2.1 Navio2-lentokontrolleri

Navio2 on Emlid-yhtiön kehittämä vapaan lähdekoodin ohjelmistoa ja ROS-järjestelmää tukeva lentokontrolleri. Navio2 kytketään piikkirimansa välityksellä Raspberry Pi-alustaan, joka käsittelee vastaanotetun ja lähtevän tiedon.

Navio2 painaa 23 grammaa ja on mitoiltaan 55mm x 65mm. Järjestelmä on laajennettavissa UART-, I2C- ja ADC-porttien välityksellä. Navio2 on toteutettu kahdella IMU-piirillä tarkkuuden parantamiseksi ja kontrollerin ilmanpainemittari pystyy selvittämään koneen korkeuden 10 cm tarkkuudella. (Emlid. Navio2.)

4.3 Prosessointi

Raspberry Pi 3 on kolmannen sukupolven Raspberry Pi. Se korvasi edeltäjänsä: Raspberry Pi 2 Model B:n helmikuussa 2016. Verrattuna edeltäjäänsä Raspberry Pi 3 sisältää 1,2 GHz 64-bittisen neliydin ARMv8-prosessorin, 802.11n langattoman paikallisverkkoadapterin, Bluetooth 4.1 –adapterin sekä Bluetooth Low Energy (BLE)-moduulin.

Laite sisältää myös edeltäjänsä tapaan 4 USB-porttia, 40 GPIO-pinniä, HDMI-portin, ethernet-portin, yhdistetyn 3,5 mm audiojakin ja komposiittivideon, kamerakäyttöliittymän (CSI), näyttökäyttöliittymän (DSI), microSD-korttipaikan ja VideoCore IV 3D-grafiikkaytimen. (Raspberry Pi 3 Model B.)

Raspberry Pi 3 on täysin taaksepäin yhteensopiva aiempien Raspberry Pi tuotesarjan mallien kanssa.

5 OHJELMOINTI

Järjestelmä ohjelmoidaan ROS-käyttöjärjestelmää apuna käyttäen. Tarkoitus on saada ROS ymmärtämään LiDAR-yksiköltä saatavaa tietoa ja ohjaamaan se edelleen autonomisen järjestelmän käyttöön.

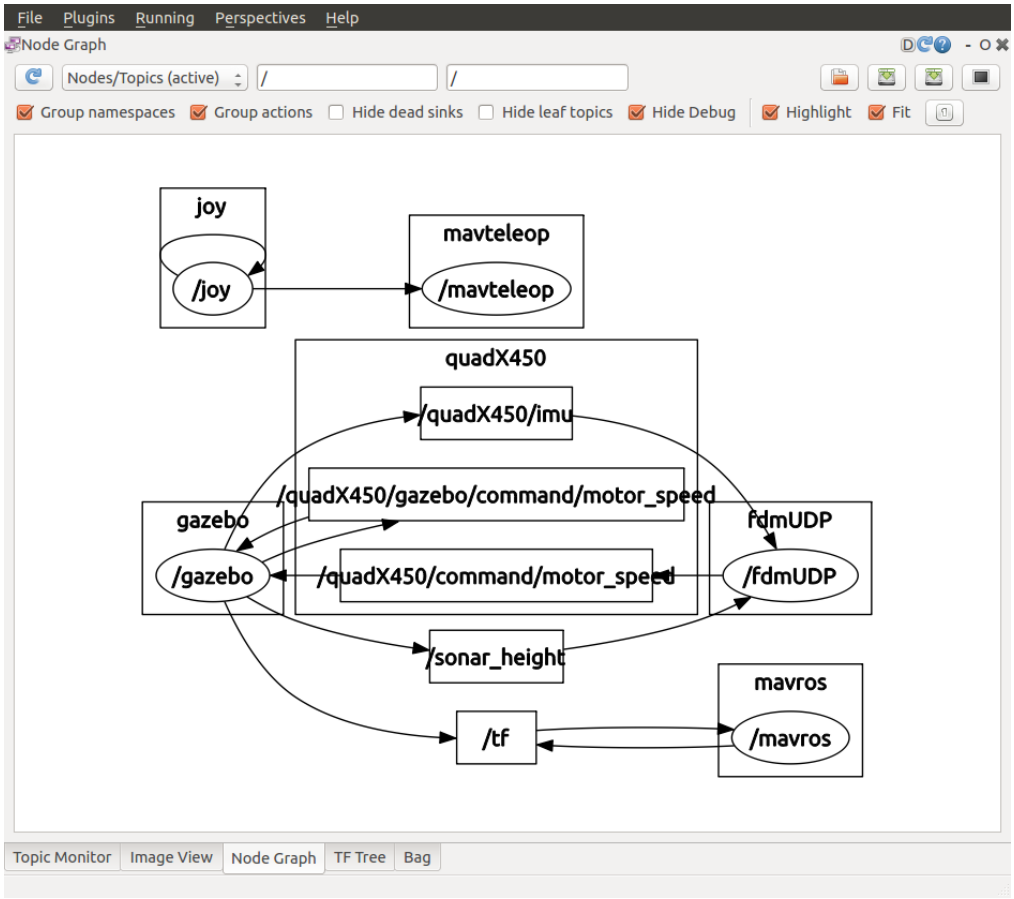
5.1 ROS-käyttöjärjestelmä

ROS eli Robotic Operating System on debianin päälle asennettava käyttöjärjestelmä, johon on mahdollista asentaa eri laitevalmistajien paketteja robotin käyttötarkoituksesta riippuen. Kaikkia tässä työssä esiteltyjä laseretäisyysmittareita on mahdollista käyttää ROS-järjestelmässä. ROS-moduulien ohjelmointikielinä toimii C++ ja Python.

ROS-kielessä on kolme termiä joihin tulee tutustua tarkemmin yhteisen sanaston saavuttamiseksi. *Node* on ROS-moduulin osa, joka lähettää tai vastaanottaa tietoa. *Topic* on noden muodostama kanava, jonka välityksellä toinen node voi muodostaa siihen jatkuvan yhteyden. *Service* on palvelu, joka pyytää hetkittäin dataa joltain nodelta.

Esimerkiksi karttapalvelun käyttöönottamiseksi ei kannata muodostaa topic-yhteyttä kahden node’in välille. Järkevämpää tällaisessa tapauksessa on muodostaa service-yhteys karttapalveluun ja vastaanottaa vain pieni määrää karttaa kerralla.

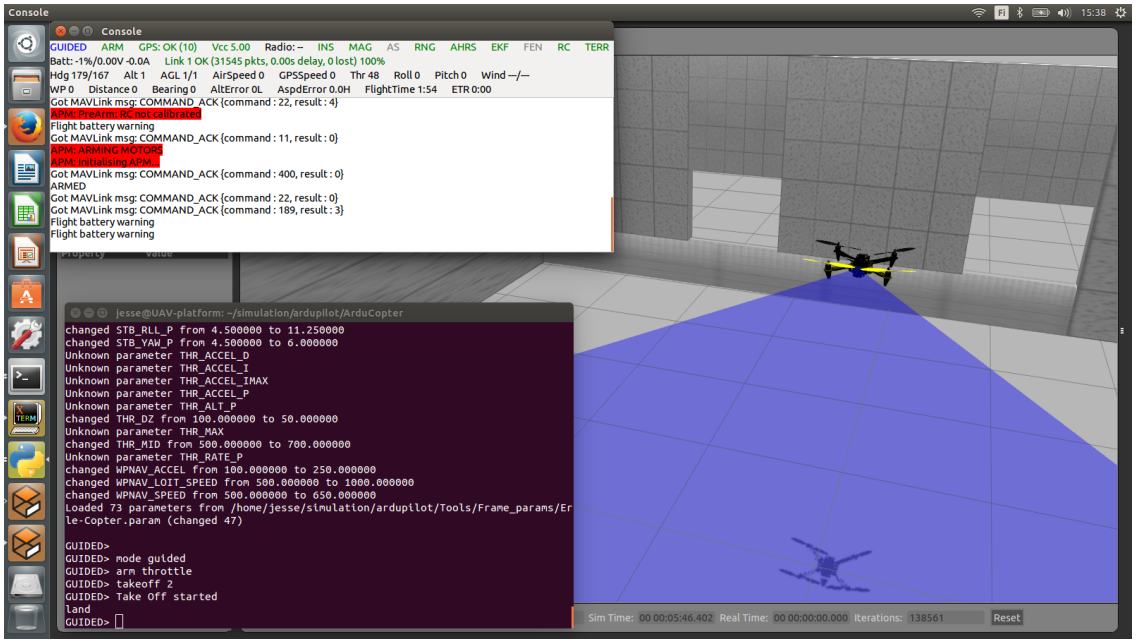
Kuviossa 1 on mallinnettu Arducopter SITL-simulaatiossa toimiva node'in verkko. Kuviossa neliöllä eristetyt alueet kuvaavat nodeja, ympyrällä eristetyt alueet servicejä ja / -symbolilla alkavat rivit edustavat topiceja. Nuolen suunnalla on merkitään datan kulkusuuntaa.



KUVIO 1. SITL-simulaation yhteyskaavio

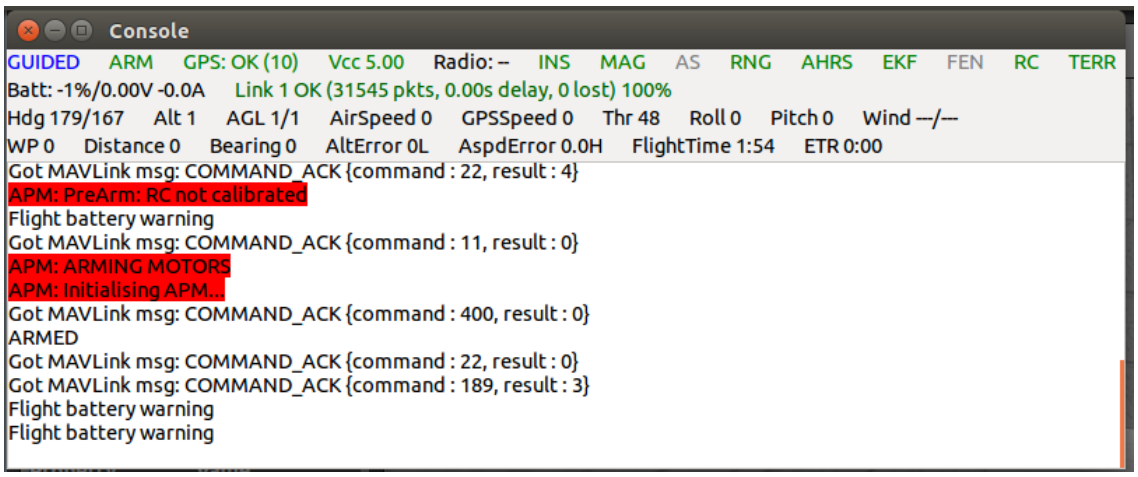
6 TESTAUS

Testaus toteutettiin Erle-Robotics –tiimin kokoamalla simulaattorilla, jonka pohjana on käytetty Arducopter SITL-simulaatiota. Käyttöjärjestelmänä Linux Ubuntu 14.04.2 64-bit ja ROS indigo. Simulaatio vastaa mittasuhteiltaan reaalista maailmaa ja simulaatio käyttää todellisia ROS-moduuleja, joten nelikopterin syötetyn koodin ja järjestelmän toimintaan voidaan luottaa. Simulaatiossa käytetty nelikopteri on Erle-Copter. Simulaation käyttöönoton komennot on esitetty liitteessä 1.



KUVA 12. Nelikopterin leijunta simulaatiossa

Simulaatiossa nelikopteria ohjataan MAVROS-komentoja käyttäen. MAVROS-paketti tarjoaa UDP MAVLink-sillan komentokeskuksen ja nelikopterin väliin. Nelikopterin toimintaa voi samanaikaisesti tarkastella konsolinäkymästä, jossa ilmoitetaan lennolle tärkeitä tietoja linkkien toimivuudesta ja nelikopterin sekä siihen liitettyjen antureiden tiloista (kuva 13).

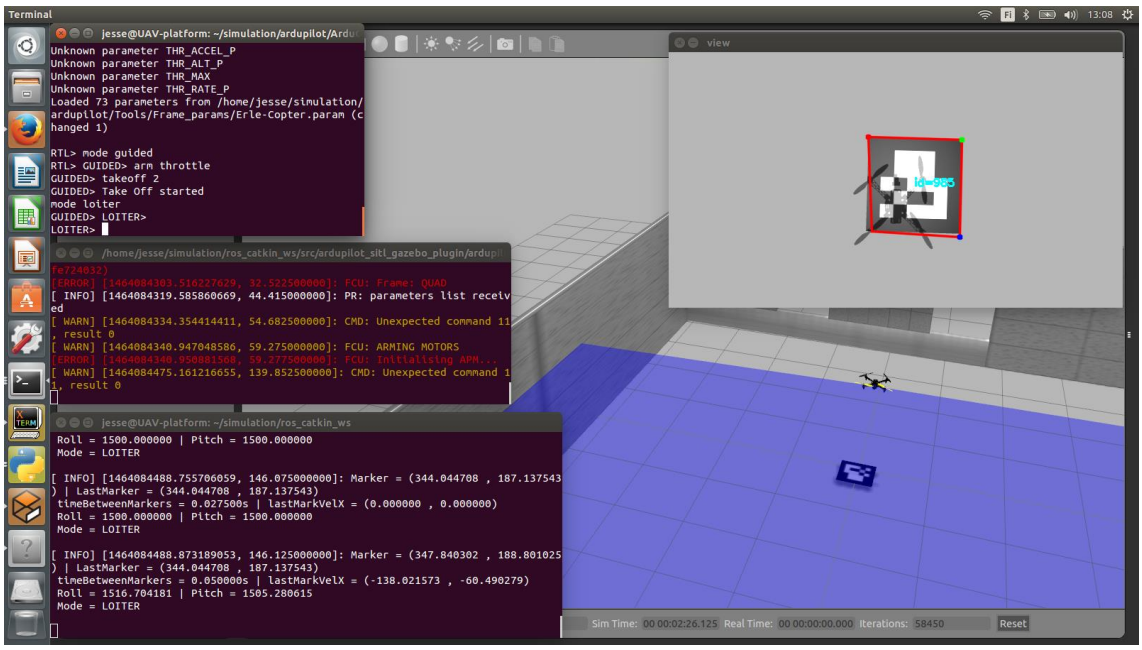


KUVA 13. MAVROS-konsolinäkymä lennon aikana

Nelikopteri renderoidaan gazebo-järjestelmässä, jossa on mahdollista mallintaa erilaisia ympäristöjä. Simulaatiossa käytetty malli on toimistorakennuksen sisätila ilman kattoa. Koska nelikopteri ja rakennus ovat mittasuhteiltaan oikeita, voidaan nelikopterin etu- tai ala-anturia hyödyntämällä luoda kaikuluotausalue.

Simulaatiossa käytetty nelikopteri on varustettu etu- ja pohja-anturilla. Pohja-anturia käytetään korkeuden kaikuluotaukseen. Etuanturina käytetään LiDAR-anturia, jonka muodostaman keilan muoto, sekä päivitysnopeuden muutos ovat muokattavissa simulaation tiedostossa erlecopter.xacro.
Koska kyseessä on keilamainen renderointi, nelikopterin eteen luotiin puolikkaan TeraRanger Tower-järjestelmän tuottama luotausalue. Vakioluotausalue on merkitty kuvassa 12 ja laajennettu luotausalue kuvassa 14 sinisellä.

Simulaatioon lisättiin merkkiseuraus-moduuli (liittessä 2), joka edustaa autonomista käytöstä simulaatiossa. Gazebo-ympäristöön on mahdollista luoda Aruco-*tag*, jonka nelikopteri tunnistaa pohjassa olevalla kamerallaan, asettuu sen päälle itsestään ja jää leijumaan esiasetettuun korkeuteen. Nelikopterin etenemistä on mahdollista seurata yht'aikaisesti sekä gazebo- että pohjakameranäkymästä.



KUVA 14. Nelikopteri leijuu Aruco-tagin päällä

7 POHDINTA

Lähtökohtana opinnäytetyölle oli suunnitella toimiva autonominen kaivoksia mallintava UAV. Tässä työssä rakennettu simulointiympäristö todentaa tarpeeksi tarkasti nelikopterin liikkuvuuden ja mahdollistaa sen edelleen kehittämisen ROS-ympäristössä.

Nykyisen merkkiseuranta-moduulin voi tulevaisuudessa korvata autonomiaa edistävillä SLAM-moduuleilla, joka vie projektia lähemmäs toteutumistaan. Darmstadt yliopistossa kehitettyä hector-SLAM-moduulia on käytetty jo muita tekniikoita hyödyntävissä kokeellisissa autonomisissa nelikoptereissa hyvin tuloksin.

Myös halvat LiDAR-yksiköt tekevät vasta tuloaan markkinoille, eikä tarjonnassa ole juuri varaa valita. Nykyisellään halvat, pyörivät LiDAR-yksiköt eivät välttämättä ole paras ratkaisu nelikopterin SLAM-järjestelmän toteuttamiseen. Kevyet, paikallaan pysyvät anturit ovat luotettavampi ratkaisu nopeamman päivitysnopeutensa ansiosta.

LÄHTEET

Hokuyo. UTM-30LX. Luettu 25.3.2016
https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html

Velodyne. HDL-64E. Luettu 25.3.2016
http://velodynelidar.com/hdl-64e.html

Velodyne. Puck LITE. Luettu 25.3.2016
http://velodynelidar.com/vlp-16-lite.html

Youtube. CSIRO – Mapping the Northparkes Mine. Luettu 20.2.2016
https://www.youtube.com/watch?v=QQeJ1xd_sOU

CSIRO. Hovermap – Autonomous Systems. Luettu 6.4.2016
https://wiki.csiro.au/display/ASL/HoverMap

GeoSLAM. Zeb1. Luettu 10.3.2016
http://geoslam.com/hardware-products/zeb1/

GeoSLAM Zeb-Revo. Luettu 10.3.2016
http://geoslam.com/hardware-products/zeb-revo/

UAV Experts. This Dramatic High-Speed Drone Chase Is Actually A Demo Of MIT's Self-Flying UAV. Luettu 21.2.2016
http://www.uavexpertnews.com/mits-self-flying-uav/

Youtube. Drone Autonomously Avoiding Obstacles at 30 MPH. Luettu 20.2.2016
https://www.youtube.com/watch?v=_qah8oIzCwk

Youtube. Autonomous UAV Project. Luettu 20.2.2016
https://www.youtube.com/watch?v=XlXc1ci40Bg

Teraranger. Technology advantages. Luettu 27.3.2016
http://www.teraranger.com/technology/advantages/

Teraranger. TeraRanger One. Luettu 25.4.2016
http://www.teraranger.com/products/teraranger-one/

Teraranger. TeraRanger Tower. Luettu 25.4.2016
http://www.teraranger.com/teraranger-tower/

Tom's Hardware. Multi-Rotors And The Hardware You Need. Luettu 25.3.2016
http://www.tomshardware.com/reviews/multi-rotor-quadcopter-fpv,3828-2.html

Emlid. Navio2. Luettu 25.3.2016
http://www.emlid.com/

Raspberry Pi 3 Model B. Luettu 18.4.2016
https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

LIITTEET

Liite 1. Simulaattorin käyttöönotto

```
sudo apt-get update
sudo apt-get install gawk make git curl cmake

sudo apt-get install g++ python-pip python-matplotlib python-serial python-wxgtk2.8 python-scipy python-opencv python-numpy python-pyparsing ccache realpath libopencv-dev

sudo pip2 install pymavlink MAVProxy catkin_pkg –upgrade

cd ~/Downloads
tar -xvzf aruco-1.3.0.tgz
cd aruco-1.3.0/
mkdir build && cd build
cmake ..
make
sudo make install

mkdir -p ~/simulation; cd ~/simulation
git clone https://github.com/erlerobot/ardupilot
cd ardupilot
git checkout gazebo

cd ~/simulation
git clone git://github.com/tridge/jsbsim.git
# Additional dependencies required
sudo apt-get install libtool automake autoconf libexpat1-dev
cd jsbsim
./autogen.sh --enable-libraries
make -j2
sudo make install

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver hkp://pool.sks-keyservers.net --recv-key 0xB01FA116
sudo apt-get update

sudo apt-get install ros-indigo-ros-base

sudo rosdep init
rosdep update

echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

```
sudo apt-get    install python-rosinstall
                ros-indigo-octomap-msgs
                ros-indigo-joy
                ros-indigo-geodesy
                ros-indigo-octomap-ros
                ros-indigo-mavlink
                unzip

sudo sh -c 'echo "deb http://packages.osrfoundation.org/drc/ubuntu $(lsb_release -sc)
main" > /etc/apt/sources.list.d/drc-latest.list'
wget http://packages.osrfoundation.org/drc.key -O - | sudo apt-key add -
sudo apt-get update
sudo apt-get install drcsim

mkdir -p ~/simulation/ros_catkin_ws/src

cd ~/simulation/ros_catkin_ws/src
catkin_init_workspace
cd ~/simulation/ros_catkin_ws
catkin_make
source devel/setup.bash

cd src/
git clone https://github.com/erlerobot/ardupilot_sitl_gazebo_plugin
git clone https://github.com/tu-darmstadt-ros-pkg/hector_gazebo/

# Rotors simulation
 git clone https://github.com/erlerobot/rotors_simulator -b sonar_plugin

## mav comm
git clone https://github.com/PX4/mav_comm.git

## glog catkin
git clone https://github.com/ethz-asl/glog_catkin.git

## catkin simple
git clone https://github.com/catkin/catkin_simple.git

# Installation of `mavros` from its source code:
cd ~/simulation/ros_catkin_ws
wstool init src     # (if not already initialized)
wstool set -t src mavros --git https://github.com/erlerobot/mavros.git
wstool update -t src

cd ~/simulation/ros_catkin_ws
catkin_make --pkg mav_msgs
source devel/setup.bash
catkin_make

mkdir -p ~/.gazebo/models
git clone https://github.com/erlerobot/erle_gazebo_models
mv erle_gazebo_models/* ~/.gazebo/models
```

Liite 2. Merkkiseuraaja

Asennus:

```
cd ~/simulation/ros_catkin_ws/src
git clone https://github.com/IkerZamora/ros_erle_pattern_follower.git
cd ~/simulation/ros_catkin_ws
catkin_make --pkg ros_erle_pattern_follower
```

Koodi:

```
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <opencv2/highgui/highgui.hpp>
#include <cv_bridge/cv_bridge.h>
#include <aruco/aruco.h>
#include <iostream>
#include <mavros/OverrideRCIn.h>
#include <mavros/State.h>

#define FACTOR  0.6

#define MINRC   1100
#define BASERC  1500
#define MAXRC   1900

// Subscriber to bottom camera
image_transport::Subscriber sub;

// Subscriber to flight mode
ros::Subscriber mavros_state_sub;

// RC publisher
ros::Publisher pub;
```

```
// Time control
ros::Time lastTime;

// Mark info
float MarkX, MarkY; // Mark center
float lastMarkX, lastMarkY; // Last mark center
double lastMarkVelX, lastMarkVelY; // Last mark velocity

//Image center
float ImageX, ImageY;

double Roll, Pitch;

// Flight mode
std::string mode;
bool guided;
bool armed;

void imageCallback(const sensor_msgs::ImageConstPtr& msg)
{
   try
   {
      // Time since last call
      double timeBetweenMarkers = (ros::Time::now() - lastTime).toSec();
      lastTime = ros::Time::now();

      char tab2[1024];
      strncpy(tab2, mode.c_str(), sizeof(tab2));
      tab2[sizeof(tab2) - 1] = 0;
      ROS_INFO("Marker = (%f , %f) | LastMarker = (%f , %f) \n timeBetweenMarkers
= %fs | lastMarkVelX = (%f , %f)\n Roll = %f | Pitch = %f\n Mode = %s \n", MarkX,
MarkY, lastMarkX, lastMarkY, timeBetweenMarkers, lastMarkVelX, lastMarkVelY,
Roll, Pitch, tab2);
```

```
aruco::MarkerDetector MDetector;
vector<aruco::Marker> Markers;
cv::Point2f MarkCenter;

// Get the msg image
cv::Mat InImage;
InImage = cv_bridge::toCvShare(msg, "bgr8")->image;

// Error between Image and Mark
float ErX = 0.0;
float ErY = 0.0;

// Get the Image center
ImageX = InImage.cols / 2.0f;
ImageY = InImage.rows / 2.0f;

// Detect markers
MDetector.detect(InImage,Markers);

// Create RC msg
mavros::OverrideRCIn msg;

lastMarkX = MarkX;
lastMarkY = MarkY;

// For each marker, draw info ant its coundaries in the image
for (unsigned int i = 0; i<Markers.size(); i++){
   Markers[i].draw(InImage,cv::Scalar(0,0,255),2);

   // Calculate the error between Image center and Mark center
   MarkCenter = Markers[i].getCenter();
   MarkX = MarkCenter.x;
   MarkY = MarkCenter.y;
   ErX = ImageX - MarkX;
   ErY = ImageY - MarkY;
```

```
}

// Calculate velocity
if (timeBetweenMarkers < 1.0){
   lastMarkVelX = (lastMarkX - MarkX)/timeBetweenMarkers;
   lastMarkVelY = (lastMarkY - MarkY)/timeBetweenMarkers;
} else{
   lastMarkVelX = 0.0;
   lastMarkVelY = 0.0;
}

// Calculate Roll and Pitch depending on the mode
if (mode == "LOITER"){
   Roll = BASERC - ErX * FACTOR;
   Pitch = BASERC - ErY * FACTOR;
}else if (mode == "ALT_HOLD"){
   Roll = BASERC - (0.5*ErX+0.1*lastMarkVelX);
   Pitch = BASERC - (0.5*ErY+0.1*lastMarkVelY);
}else{
   Roll = BASERC;
   Pitch = BASERC;
}

// Limit the Roll
if (Roll > MAXRC)
{
   Roll = MAXRC;
} else if (Roll < MINRC)
{
   Roll = MINRC;
}

// Limit the Pitch
if (Pitch > MAXRC)
{
```

```
            Pitch = MAXRC;
        } else if (Pitch < MINRC)
        {
            Pitch = MINRC;
        }

        msg.channels[0] = Roll;     //Roll
        msg.channels[1] = Pitch;    //Pitch
        msg.channels[2] = BASERC;   //Throttle
        msg.channels[3] = 0;        //Yaw
        msg.channels[4] = 0;
        msg.channels[5] = 0;
        msg.channels[6] = 0;
        msg.channels[7] = 0;

        pub.publish(msg);

        cv::imshow("view", InImage);
        cv::waitKey(30);
    }
    catch (cv_bridge::Exception& e)
    {
        ROS_ERROR("Could not convert from '%s' to 'bgr8'.", msg->encoding.c_str());
    }
}

void mavrosStateCb(const mavros::StateConstPtr &msg)
{
    if(msg->mode == std::string("CMODE(0)"))
        return;
    //ROS_INFO("I heard: [%s] [%d] [%d]", msg->mode.c_str(), msg->armed, msg->guided);
    mode = msg->mode;
    guided = msg->guided==128;
```

```
    armed = msg->armed==128;
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "image_listener");
    ros::NodeHandle nh;
    lastTime = ros::Time::now();
    image_transport::ImageTransport it(nh);
    sub = it.subscribe("/erlecopter/bottom/image_raw", 1, imageCallback);
    mavros_state_sub = nh.subscribe("/mavros/state", 1, mavrosStateCb);
    pub = nh.advertise<mavros::OverrideRCIn>("/mavros/rc/override", 10);;
    ros::spin();
}
```