Mars-luotainten reitinhakualgoritmit ja visuaalinen paikantaminen

Jerry Mesimäki

Kandidaatintutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 19. marraskuuta 2014

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Jerry Mesimäki

Työn nimi — Arbetets titel — Title

Mars-luotainten reitinhakualgoritmit ja visuaalinen paikantaminen

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kandidaatintutkielma	19. marraskuuta 2014	15

Tiivistelmä — Referat — Abstract

Tiivistelmä.

Avainsanat — Nyckelord — Keywords
avainsana 1, avainsana 2, avainsana 3

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

Ihmiskunta on ollut kiinnostunut avaruudesta ja erityisesti lähiplaneetoistaan jo antiikin ajoista asti. Tähtitieteen historiaan mahtuu useita merkkitäviä läpimurtoja kuten aurinkokeskeisen maailmankuvan ymmärtäminen ja vuoden 1969 Apollo-ohjelman miehitetty lento Kuuhun. Intressit avaruuden tutkimiselle kasvavat jatkuvasti ja viimevuosina myös kaupalliset tahot ovat luoneet työllisyyttä esimerkiksi avaruusturismin avulla. Jotkin yritykset tutkailevat jo mahdollisuuksia kaivaa kallisarvoisia mineraaleja planeettaamme kiertävistä asteroideista. Kaikki tämä vaatii useiden eri tieteenalojen yhteistyötä, jotta teknologiat kehittyisivät vastaamaan avaruuden asettamia haasteita.

Viimeisten vuosikymmenten aikana keskeisiksi tutkimusvälineiksi aurinkokuntamme kartoittamisessa ovat muodostuneet miehittämättömät avaruusluotaimet, joiden avulla tiedemiehet ovat keränneet huomattavia määriä kiinnostavaa dataa Maan naapureista. Tänä päivänä luotaimia löytyy jo useimpien planeettojen kiertoradoilta, Venuksen ja Marsin pinnoilta sekä tuoreimpien arvioiden mukaan Aurinkokunnan ulkopuolelta.

Kosmisessa mittakaavassa etäisyydet ovat aivan omaa luokkaansa ja tästä syystä luotainten ohjaaminen Maasta käsin on useimmissa tapauksissa hidasta. Asiaa hankaloittaa myös se ettei reaaliaikaisen kuvan saaminen ole mahdollista vaikka radioaallot kulkevat valonnopeudella halki avaruuden. Tästä johtuen tiedemiehet ovat pyrkineet automatisoimaan erilaisin algoritmein luotaintensa toimintaa. Tekoälyä taas käytetään, jotta laite kykenisi suorittamaan sille annetut tehtävät mahdollisimman itsenäisesti. Esimerkiksi Mars-mönkijät pyrkivät väistelemään maastosta löytyviä vaaroja kuvaamalla omaa ympäristöään ja analysoimalla sen perusteella itselleen suotuisimpia reittejä.

Uusimpien mönkijöiden kohdalla on jo mahdollista päivittää näiden ohjelmistoa vaikka ne sijaitsisivat toisella planeetalla. Siksi onkin oleellista jatkaa niissä käytettyjen ohjelmistojen kehittämistä vielä laukaisun jälkeen, jolloin laitteistojen potentiaalinen hyöty kasvaa ja tarve uusien laitteiden lähettämiselle avaruuteen pienenee.

2 MER eli Mars Exploration Rover

Marsiin lähetettiin vuonna 2003 Nasan toimesta kaksi MER-laitetta (Mars Explorarion Rover, myöhemmin mönkijä) – Spirit sekä Rover – etsimään merkkejä veden esiintymisestä planeetan pinnalta. Yksi ongelmista on mönkijöiden liikuttelu maasta käsin 26 minuuttia kestävän signaalin siirtymisestä aiheutuvan viiveen takia. Tyypillisesti laitteille lähetetään komentoja vain aamuisin kerran Marsin vuorokaudessa (myöhemmin kierto), jotka ne toteuttavat kierron aikana. Ennen iltaa niiden keräämä data taas lähetetään Maahan analysoitavaksi. Tästä johtuen mönkijöiden riittävä autonomia on kriittistä, jotta ne kykenisivät liikkumaan mahdollisimman nopeasti tieteellisesti kiinnostavien kohteiden välillä.

Mönkijöitä voidaan liikuttaa kahdella eri tavalla: sokkoajona, jossa Maahan saapuneista kuvista päätellään mihin suuntaan on turvallista ajaa ja ajaminen suoritetaan vaaroista välittämättä haluttuun kohteeseen. Toinen vaihtoehto on luovuttaa vastuu ajamisesta mönkijöissä sijaitsevalle AutoNav-järjestelmälle, joka pyrkii liikkumaan haluttuun kohteeseen ottaen kuitenkin huomioon ympäristössä sijaitsevat vaarat ja väistämään niitä. Useimmiten näitä käytetään yhdessä siten, että sokkoajoa sovelletaan niin pitkälle kuin saaduista kuvista päätellen on turvallista ja tämän jälkeen AutoNav jatkaa ajamista.

Toisinaan automaatio ei toimi ja tämä estää mönkijöitä saavuttamasta kohteitaan. Heinäkuussa 2006 molemmat saivat järjestelmäpäivityksen mukana korjauksia em. ongelmaan, joita tämä essee pääasiassa käsittelee.

2.1 Mönkijöille asetetut tavoitteet

2.2 Laitteisto

2.3 AutoNav

AutoNav perustuu GESTALT-algoritmiin (grid-based estimation of surface traversability applied to local terrain), jossa mönkijän stereokameroiden ottamista kuvista luodaan malli tämän välittömästä ympäristöstä. Yksi osa mallista on ruudukkopohjainen hyvyyskartta. Kartalla jokainen solu saa sen ajettavuutta kuvaava hyvyysarvon. Hankala maasto tuottaa matalan arvon

ja helppo maasto korkean. Maastoa, jota mönkijä ei voi läpäistä levitetään kartalla mönkijän koon verran, jolloin laite voidaan sijoittaa pisteeksi kartalle laskentaa varten.

Kartoituksen jälkeen järjestelmä luo joukon kaaria, joita pitkin kulkemalla voidaan saavuttaa haluttu kohde. Mikäli kaari ei ole suora niin sille lasketaan myös kääntymispisteet. Jokaista kaarta arvioidaan kolmella kriteerillä: vaarojen välttely, kääntymisajan minimointi ja kohteen saavuttaminen. Arvion perusteella sovellus järjestää kaarien välille äänestyksen siten, että hyvyyskartalla turvallisille kaarille annetaan enemmän ääniä kuin turvattomille, kääntymispisteiden lukumäärä korreloi negatiivisesti annettujen äänien kanssa ja mönkijää lähemmäksi kohti kohdetta vievät kaaret saavat enemmän ääniä. Äänet lasketaan tämän jälkeen yhteen ja mönkijä ajaa pienen ennalta määrätyn matkan voittanutta kaarta. Edellä selitettyä prosessia toistetaan kunnes kohde saavutetaan, ennalta määrätty tavoiteaika ylittyy tai tapahtuu virhe.

Ongelmaksi MER-laitteiden järjestelmän ensimmäisessä versiossa muodostui tilanne, jossa mönkijä kohtasi riittävän ison kiviröykkiön. Sen kiertämiseksi olisi jouduttu valitsemaan sellainen kaari, joka ei vie mönkijää riittävän suoraan kohti kohdetta. Suoremmat kaaret taas kulkivat kivien yli. Tilanne aiheutti äänestysvaiheessa konfliktin, jossa vaaran välttäminen oli liian suuressa konfliktissa suorimman reitin haun kanssa. Järjestelmä ei antanut mönkijän liikkua mihinkään suuntaan.

2.4 GESTALT ennen globaalia reitinsuunnittelua

2.5 Päivitetty GESTALT

3 Field D* ja lineaarinen interpolointi

3.1 Aikaisemmat algoritmit

Useimmissa mobiilirobotiikan navigaatiosovelluksissa ympäristöä mallinnetaan ruudukkona, jossa jokainen ruutu saa arvokseen jonkin arvion sen kuljettavuudesta. Tällä tavoin robotti voi tarvittaessa väistää ympäristössään sijaitsevia esteitä tai liian vaaralliseksi koettuja alueita. Ruudukon lisäksi yleisesti käytetyt algoritmit generoivat myös verkon, jonka solmut

sijoitetaan keskelle jokaista ruutua. Verkon kaaret muodostetaan ruudussa sijaitsevan solmun ja tämän naapuriruutuihin asetettujen solmujen välille. Tässä mallissa reitinhaku verkossa voidaan toteuttaa esimerkiksi Anthony Stentzin vuonna 1995 esittelemän D*-algoritmin avulla [Ste95]. Pääasialliseksi ongelmaksi jää kuitenkin em. kyky löytyää vain sellaiset reitit, joissa robotti liikkuu $\pi/4$ käännöksillä ruutujen välillä.

3.2 Kustannusarvion parantaminen interpoloinnin avulla

Algoritmin perustana toimii metodi, jossa jokaisesta ruudukossa sijaitsevasta solmusta lasketaan halvin mahdollinen kustannusarvio haluttuun kohdepisteeseen [FS07]. Perinteisesti ruudukkopohjaisessa reitinsuunnittelussa on käytetty seuraavaa kaavaa:

$$g(s) = \min_{s' \in nbrs(s)} (c(s, s') + g(s')),$$

missä $nbrs(s)$ on joukko kaikista solmun s naapureista, $c(s, s')$ on kustannus kulkemiseen kaaren s ja s' välillä, sekä $g(s')$ on kustannusarvio solmulle s'.

Kaavassa oletetaan, että solmusta s voidaan siirtyä tämän naapureihin ainoastaan suoraa linjaa pitkin, joka taas johtaa aikaisemmin mainittuun ongelmaan muodostaa parhaita mahdollisia reittejä robotin rajoittuneen suuntaamisen takia. Tämä voitaisiin korjata sijoittamalla s':n tilalle s_b, jossa s_b on mikä tahansa piste solmuun liittyvän ruudun reunalla. Näitä pisteitä on kuitenkin ääretön määrä, joka tekee jokaisen pisteen laskennasta mahdotonta.

Muokkaamalla verkkoa voimme silti muodostaa approksimaation jokaiselle pisteelle s_b käyttäen lineaarista interpolointia. Sen sijaan, että solmu sijoitettaisiin ruudun keskelle, asetetaan solmu jokaisen ruudun kulmaan ja täten kaaret kulkevat ruudukon reunoja pitkin. Nyt yhden kaaren kustannus voidaan valita siten, että se on pienempi niistä kahdesta ruudusta joiden välillä se kulkee.

Tämä muokkaus johtaa siihen, että paras mahdollinen reitti kulkee jotkin kaksi naapurisolmua $\overrightarrow{s_1 s_2}$ yhdistävän kaaren läpi. Solmulle s voidaan nyt laskea kustannusarvio, kun reitti kulkee sen ruudun läpi em. kaarelle. Laskemista varten tarvitaan arviot solmujen $s1$ ja $s2$ sekä keskimmäisen ruudun c ja alemman ruudun b kustannuksista.

Kustannusarvion tuottamiseen käytetään vielä oletusta, että kustannusarvio mille tahansa pisteelle s_y, joka sijaitsee kaarella $\overrightarrow{s_1 s_2}$, on funktioiden $g(s1)$ ja $g(s2)$ lineaarikombinaatio:

$$g(s_y) = yg(s_2) + (1 - y)g(s_1),$$

missä y on etäisyys s_1:stä s_y:hyn. Tulee kuitenkin huomata, että s_y ei välttämättä ole em. funktioiden lineaarikombinaatio, mutta tämän oletuksen tuottama approksimaatio toimii käytännässö riittävän hyvin kun halutaan muodostaa suljettu muoto solmun s kustannusarvion palauttavalle funktiolle.

Approksimaation perusteella solmun s kustannus kun tiedetään $s1$, $s2$, ruutukustannukset c ja b voidaan laskea seuraavasti:

$$\min_{x,y}[bx + c\sqrt{(1 - x)^2 + y^2} + g(s_2)y + g(s_1)(1 - y)],$$

missä $x \in [0, 1]$ on solmusta s alareunaa pitkin kuljettu matka kunnes käännytään ruudun yli kohti oikeaa reunaa pisteeseen, joka on $y \in [0, 1]$ etäisyyden päässä solmusta s_1.

Tehdään vielä oletus, että (x^*, y^*) ovat x:n ja y:n arvot, joilla ylläoleva kaava saadaan ratkaistua. Lineaarisen interpoloinnin johdosta toinen arvoista on joko yksi tai nolla. Mikäli kustannus liikkua ruudun c yli on pienempi kuin ruudun reunoja pitkin kulkeminen niin halvin reitti halkaisee ruudun c ja täten joko $x^* = 0$ tai $y^* = 1$. Jos taas polku ei halkaise ruutua c niin $y^* = 0$. Täten polku on kulkee solmusta s suoraan alareunaa pitkin kohtia solmua $s1$, siirtyy jonkin matkan x alareunalla ja leikkaa tämän jälkeen ruudun halki suoraan solmuun $s2$, tai halkaisee ruudun c kulkemalla suoraan solmusta s johonkin oikean reunan pisteeseen s_y. Halvin polku riippuu c:n ja b:n koosta, sekä s_1:n ja s_2:n kustannuserosta $f = g(s_1) - g(s_2)$. Mikäli $f < 0$ niin paras mahdollinen polku on 1. tapaus, jos taas $f = b$ niin polun kustannus kulkien jonkin matkan alareunaa on yhtäpitävä sen kanssa, että alareunaa ei kuljeta ollenkaan. Jälkimmäisestä polusta voidaan ratkaista kustannuksen minimoiva y seuraavasti.

Olkoon $k = f = b$. Kustannus solmusta s kaaren $\overrightarrow{s_1 s_2}$ lävitse on

$$c\sqrt{1 + y^2} + k(1 - y) + g(s2).$$

Jossa kustannuksen derivaatasta suhteessa y:hyn ja asettamalla se nollaksi saadaan

$$y^* = \sqrt{\frac{k^2}{(c^2 - k^2)}}$$

Lopputulos on sama huolimatta siitä kuljetaanko alareunaa pitkin, joten merkitseväksi tekijäksi jää se kumpaa reunaa kulkeminen tulee halvemmaksi. Mikäli $f < b$ niin käytetään oikeaa reunaa ja lasketaan polun kustannus arvolla $k = f$. Jos taas $b < f$ niin käytetään alareunaa jolloin $k = b$ ja $y^* = 1 - x^*$. Täten algoritmi halvimman polun laskemiseen solmusta s mihin tahansa pisteeseen kaarelle, joka sijaitsee vierekkäisten naapurien s_a ja s_b välissä laskemiseen on seuraava:

```
ComputeCost(s, s_a, s_b)
    if (s_a on solmun s diagonaalinaapuri)
        s_1 = s_b;
        s_2 = s_a;
    else
        s_1 = s_a;
        s_2 = s_b;

    c on kustannus ruudulle, jonka kulmat ovat s, s_1, s_2;
    b on kustannus ruudulle, jonka kulmat ovat s, s_1, mutta ei s_2;

    if (min(c, b) = ∞)
        v_s = ∞;
    else if (g(s_1) ≤ g(s_2))
        v_s = min(c, b) + g(s_1);
    else
        f = g(s_1) − g(s_2);
        if (f ≤ b)
            if (c ≤ f)
                v_s = c√2 + g(s_2);
            else
                y = min(f / √(c^2 − f^2), 1);
                v_s = c√(1 + y^2) + f(1 − y) + g(s_2);
        else
            if(c ≤ b)
                v_s = c√2 + g(s_2);
            else
                x = 1 − min(b / √(c^2 − b^2), 1);
                v_s = c√(1 + (1 − x)^2) + bx + g(s_2);
    return v_s;
```

3.3 Field D*

Seuraava algoritmi on optimoimaton esimerkki Field D*:in toteutuksesta [FS07], joka pohjautuu aikaisempaan D* Lite-algoritmiin [KL02]. Sen tarkoituksena on koota ylempänä esitellyn interpoloinnin avulla lasketut kustan-

nukset yhteen ja muodostaa edullisin reitti kahden pisteen välillä. Algoritmi on toteutettu siten, että se ottaa huomioon muutokset ympäristössä, joita voi syntyä esimerkiksi sään muuttumisen seurauksena.

```
key(s)
    return [min(g(s), rhs(s)) + h(s_start, s); min(g(s), rhs(s))];

UpdateState(s)
    if solmussa s ei ole viela kayty
        g(s) = ∞;
    if (s ≠ s_goal)
        rhs(s) = min_{(s',s'')∈connbrs(s)} ComputeCost(s, s', s'');
    if (s ∈ OPEN)
        OPEN.remove(s);
    if (g(s) ≠ rhs(s))
        OPEN.insert(key(s) : s)

ComputeShortestsPath()
    while (min_{s∈OPEN}(key(s)) < key(s_start) || rhs(s_start) ≠ g(s_start))
        OPEN.remove(smallest(s));
        if (g(s) > rhs(s))
            g(s) = rhs(s);
            for all s' ∈ nrbs(s) UpdateState(s');
        else
            g(s) = ∞;
            for all s' ∈ nbrs(s) UpdateState(s');

Main()
    g(s_start) = rhs(s_start) = ∞;
    g(s_goal) = ∞;
    OPEN.insert(key(s_goal) : s_goal)
    while true
        ComputeShortestPath();
        Odota muutoksia ruutujen kustannusarvioissa
        for x in muuttuneet_ruudut
            for s in x
                UpdateState(s);
```

Pseudokoodissa $connbrs(s)$ on joukko solmun s perättäisiä naapuripareja: $(s_1, s_2), (s_2, s_3), (s_3, s_4), (s_4, s_5), (s_5, s_6), (s_6, s_7), (s_7, s_8), (s_8, s_1)$. $g(s)$ on solmun s kustannusarvio, $rhs(s)$ kertoo arvion solmun s kustannuksesta yhden askeleen päästä, $OPEN$ toimii prioriteettijonona nousevassa järjestyksessä solmuille, joille $g(s) \neq rhs(s)$. $h(s_{start}, s)$ on heuristinen arvio kustannuksesta s_{start} solmusta solmuun s.

4 Visuaalinen odometria

Yksi mönkijöiden haasteellisimmista tehtävistä on jatkuvasti ylläpitää tietoa tämän omasta paikasta ja asennosta. Suunnitteluvaiheessa tavoitteeksi asetettiin maksimissaan kymmenen porsentin virhemarginaali sadan metrin matkalla. Tehtävää varten niihin on sijoitettu inertian mittausyksikkö, joka

yhdessä rengasodometrian kanssa pyrkii päättelemään kuinka pitkän matkan mönkijä on liikkunut ja missä asennossa se kullakin hetkellä on. Tämä menetelmä toimii hyvin tasaisella sekä hyvälaatuisella maastolla saavuttaen sille asetetun tavoitteen.

Ongelmaksi muodostuvat kuitenkin tilanteet, joissa maaston pito pienenee huomattavasti kuten esimerkiksi hiekka-aavikolla. Myös kraaterien kaltaiset kaltevat alueet tuottivat hankaluuksia inertian mittausyksikölle ylläpitää tietoa mönkijän paikasta. Näissä tilanteissa avuksi otetaan visuaalinen odometria, jossa kahdesta eri hetkenä otetusta 256x256 stereokuvaparista tunnistetaan ympäristön piirteitä ja pyritään havaittujen erojen avulla arvioimaan mönkijälle tarkemmat arvot tämän kuudelle vapausasteelle. Tämä ominaisuus kuitenkin hidastaa mönkijän liikkumista yhtä suuruusluokkaa pienemmäksi, joten se kytketään päälle ainoastaan tarvittaessa. [CMM05]

4.1 Algoritmi

Tarkoituksena on siis löytää kuvaparista sellaiset piirteet, jotka on mahdollista löytää myös lyhyen matkan jälkeen otetusta uudesta kuvaparista. Tämä mahdollistaa melko tarkan arvion tekemisen siitä kuinka paljon kuvan ottaja on liikkunut kuvien ottamisen välillä ja liikkeen suunta saadaan myös selville. Algoritmin lähtökohtana on tilastollinen menetelmä nimeltä suurimman uskottavuuden estimointi. Ensimmäisen tämänkaltaista paikantunnistusmahdollisuutta tutki Larry Matthies vuonna 1986 ja samaan työhön perustuu myös Mars-mönkijöissä käytetty algoritmi [MS87].

Menetelmä voidaan jakaa karkeasti neljään osaan, jotka suoritetaan esitellyssä järjestyksessä. Niissä sovelletaan niin digitaalista signaalinkäsittelyä kuin tilastollisia menetelmiä.

4.1.1 Piirteiden tunnistus

Ensimmäisessä vaiheessa tarkoitus on etsiä kuvaparista selkeitä piirteitä, joiden voidaan olettaa löytyvän maastosta liikkumisen jälkeen. Mönkijöiden tapauksessa on käytetty konenäön osalta tunnettua kulmien tunnistusta, jossa kuvien pikseleille annetaan kiintoisuusarvo ja korkeimman arvon saaneet valitaan kuvien kannalta kiinnostaviksi piirteiksi. On myös toivottavaa,

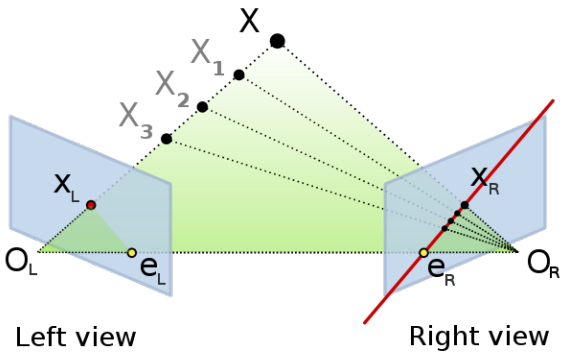
että kiinnostavia piirteitä löytyy riittävän monta, jotta ne kattaisivat tarpeeksi ison alueen kuvasta. Tämä auttaa varmistamaan sen, että kuvapareista voidaan mahdollisimman suurella todennäköisyydellä tunnistaa samat kohteet.[CMM05]

4.1.2 Piirreperusteinen syvyyden estimointi stereokuvia käyttäen

Tämän vaiheen ymmärtämisen osalta oleellista on ensin perehtyä stereokuvaparin sovitukseen (eng. stereo matching). Useille orgaanisille olennoille ympäristön hahmottaminen perustuu laajalti syvyysnäön käyttöön ja aivot prosessoivat tätä dataa suhteellisen vaivattomasti, jotta voisimme reagoida asioihin oikealla tavalla. Konenäön kannalta tämä on kuitenkin muodostunut ongelmaksi ja sen ratkaisemiseksi on kehitelty menetelmiä kuten stereokuvaparin sovittaminen. Tarkoituksena on luoda kolmiulotteinen rekonstruktio useasta eri kuvasta, jotka on otettu samasta kohteesta, mutta eri kuvakulmista. Tämä toteutetaan siten, että kuvista pyritään löytämään samoja asioita kuvaavat pikselit ja laskemaan niiden etäisyys eri kuvien välillä, jolloin voidaan luoda syvyysarvioita perustuen kameroiden sijaintiin. Suotuisa lopputulos on tiheä syvyyskartta, jossa kuvien jokaiselle pikselille on arvioitu tarkka syvyysarvo kuvaamaan pikselin paikkaa kolmiulotteisessa avaruudessa.[Hon10]

Toinen vaihe siis käyttää yllä esiteltyä menetelmää piirteiden sijainnin laskemiseen. Kameroiden hyvästä kalibroinnista johtuen sovittaminen tehdään epipolaarisella linjalla käyttäen muutaman pikselin kompensaatiopuskuria linjan ylä- ja alapuolella. Parhaan sovituksen saamiseksi kuvapariin sovelletaan Hans Moravecin väitöskirjassaan esittelemää pseudo-normalisoitua korrelaatiota, joka ottaa huomioon esimerkiksi vaihtuvat valaistusolosuhteet[Mor80]. Kolmiulotteinen malli luodaan, jotta voitaisiin varmistua myöhemmin otetun kuvaparin piirteiden vastaavan jo valittuja piirteitä. Lopuksi varsinainen sijainti saadaan tutkimalla niitä pisteitä, joissa molemmista kameroista lähtevät säteet leikkaavat yhdessä piirteiden kanssa. Erinäisistä tekijöistä johtuen säteet eivät silti välttämättä kohtaa ja niiden välistä etäisyyttä käytetään sovituksen hyvyysarvona, missä pienempi etäisyys kuvaa parempaa sovitusta.[CMM05]

Kuva 1: Vasemmasta kamerasta lähtevä suora $O_L - X$ näkyy oikealle kameralle epipolaarisena linjana $e_R - x_R$. (Kuva: Arne Nordmann, Lisenssi: GFDL)



4.1.3 Piirteiden seuraaminen

Mönkijän liikuttua noin 75 senttimetrin matkan otetaan toinen kuvapari, jonka kuvien päälle heijastetaan aikaisemmin valitut kiinnostavat piirteet rengasodometrian perusteella. Korrelaatiopohjaisella haulla samat piirteet etsitään uusista kaksiulotteisista kuvista. Nyt uudesta kuvaparista löydetyt halutut piirteet sijoitetaan kolmiulotteiseen malliin edellisen vaiheen tapaan, mutta tällä kertaa sovittaminen on huomattavasti nopeampaa, sillä piirteiden sijainti tiedetään jo edellisen kuvaparin perusteella. Mikäli joidenkin piirteiden sijainti ensimmäisessä kuvaparissa eroaa liikaa toisesta kuvaparista lasketusta sijainnista niin piirteitä ei käytetä enää tulevissa laskuissa.[CMM05]

4.1.4 Liikkeen luotettava arviointi

Mikäli rengasodometriasta saatu tieto liikutusta matkasta pitää paikkaansa niin molempien kuvaparien sovittamalla luodussa kolmiulotteisessa mallissa piirteiden sijainnit pysyvät ennaltamäärätyn virhemarginaalin sisällä eikä jatkotoimenpiteitä täten tarvita vaan mönkijä pystyy informaation avulla päivittämään sijaintinsa kartalla.

Visuaalisesta odometriasta on kuitenkin eniten hyötyä juuri silloin kun muilla tavoin ei voida saada luotettavaa sijainnista kertovaa tietoa. Tämä

havaitaan siten, että tunnistettujen piirteiden sijainti eroaa mallissa odotetusta ja eron perusteella määritellään mönkijän todellinen liike kuvaparien ottamisen välissä.

Prosessi on kaksivaiheinen, joista ensimmäisessä luodaan karkea arvio kuvaparien väliselle liikkeelle käyttäen pienimmän neliösumman menetelmää. Tähän löytyy nopea sekä luotettava suljetun muodon ratkaisu. Helpon laskettavuutensa, mutta hitaampia menetelmiä epätarkemman arvion vuoksi ensimmäisellä menetelmällä pyritään ainoastaan poistamaan poikkeavat havainnot, jolloin toinen vaihe voidaan suorittaa nopeammin. Jäljelle jäänyttä joukkoa käytetään toisessa vaiheessa tarkempaan arvioon kuljetun liikeradan laskemiselle. Epätoivotut piirteet poistetaan seuraavalla tavalla:

1. Valitaan satunnaisesti pieni määrä piirteitä ja lasketaan niiden avulla kuvaparien välinen liike soveltaen pienimmän neliösumman menetelmää.

2. Kaikki aikaisemmissa vaiheissa löydetyt piirteet heijastetaan valittujen pisteiden kanssa samaan kuvaan käyttäen hyväksi ensimmäisessä kohdassa saatua arviota liikeradasta. Jokainen satunnaisesti valittu piirre, joka osuu riittävän lähelle tätä vastaavaa heijastetta saa yhden pisteen.

3. Kahta ensimmäistä vaihetta iteroidaan haluttu määrä ja lopuksi valitaan se piirre, joka on saanut eniten pisteitä. Tätä kolmivaiheista algoritmia voidaan soveltaa kunnes saadaan tarvittava määrä piirteitä, jotka vastasivat karkeaa arviota kuljetusta liikeradasta.

Nyt prosessi on valmis tekemään tarkan arvion käyttäen tilastotieteellistä menetelmää nimeltä suurimman uskottavuuden estimointi. Arvion tekeminen tapahtuu seuraavalla tavalla:

Olkoon P_pj ja Pcj mönkijän sijainti ennen ja jälkeen liikkeen. Jolloin

$$P_{cj} = RP_{pc} + T + e_j,$$

missä R ja T ovat mönkijän rotaatio sekä translaatio ja e_j on yhdistetty virhe piirteiden havaitussa sijainnissa. Tässä arviossa kolme akselin rotaatiota θ_R ja translaatio T ovat suoraan määritelty minimoimalla eksponenttien summa:

$$\sum r_j^T W_j r_j$$

$$r_j = P_{cj} - RP_{pj} - T,$$

missä W_j on $e_j : n$ kovarianssi-käänteismatriisi. Tämän epälineaarisen ongelman minimointi on toteutettu linearisoimalla sekä iteratiivisella prosessilla. Tarkan algoritmista tekee se, että suurimman uskottavuuden estimoinnissa voidaan eliminoida rotaatiomatriisin arvioinnista syntyvä virhe laskemalla arvio akselien rotaatioille θ_R suoraan, toisin kuin pienimmän neliösumman menetelmässä. Lisäksi lopullinen arvio pitää sisällään myös tiedon kaikista rengasodometriasta saaduista virheellisistä arvioista, jonka avulla suurimman uskottavuuden estimointi on erittäin tarkka menetelmä sijainnin laskemiseksi.

[CMM05]

4.2 Algoritmin validointi Maan pinnalla

Ennen varsinaista käyttöönottoa visuaalista odometriaa kokeiltiin useilla erilaisilla alustoilla kuten esimerkiksi NASAn Jet Propulsion Laboratoryn Rocky 8 -mönkijällä. Kahdella vaarojen havaitsemiseen tarkoitetulla kameraparilla varustettuna laite muistuttaa Marsiin lähetettyjä Spiritiä sekä Opportunityä. Kokeiluun käytetty Johnson Valley tarjoaa mäkisen sekä hienon hiekan vuoksi liukkaan maaperän, jonka tarkoituksena on saada rengasodometria tuottamaan vääriä tuloksia, jolloin konenäön avulla tehdyn paikannuksen hyödyt tulevat esiin. Rockyn todellisen liikkeen sekä asennon mittaamiseen otettiin avuksi takymetri, jonka virhemarginaali pysyy alle kahdessa millimetrissä liikutulla matkalla sekä alle 0.2 asteessa asennon suhteen. Visuaalinen odometria mittasi mönkijän kulkeman matkan reilusti alle tavoitteeksi asetetun 10% virhemarginaalin ja virhettä kertyi alle 1.5%. Virhe asennon suhteen pysyi kaikissa testeissä alle viidessä asteessa.

Lähemmäksi Mars-mönkijöiden todellista laitekokoonpanoa päästiin sisätiloissa ajetulla MER Surface System Testbed Lite -järjestelmällä. Ainoa merkittävä ero on lähinnä vaarojen havaitsemiseen tarkoitetussa kameraparissa, jonka kuvissa testimönkijän näkökenttä on 120 astetta verrattuna Mars-mönkijöiden 45 asteeseen. Laitteella ajettiin seitsemän kertaa 35 senttimetrin matka ja visuaalisen odometrian sekä rengasodometrian virheet kirjattiin ylös jokaisella pysähtymiskerralla. Koko matkan ajan konenäkö pystyi pitämään virhemarginaalin alle yhdessä prosentissa, kun taas renkaista

saadut mittaukset vaihtelivat luotettavuudessaan, mutta matkan kasvaessa virhemarginaali pääasiassa kasvoi ja ylitti maksimiksi asetetun 10% arvon. [CMM05]

4.3 Visuaalinen odometria Marsin pinnalla

Spirit ja Opportunity kuvaavat ympäristöään näiden mastoihin kiinnitetyillä NAVCAM-kamerapareilla ja kameroiden tuottamat kuvaparit annetaan visuaalisen odometrian käsiteltäväksi. Paras tulos saadaan kun liike tapahtuu kerrallaan enintään 75 senttimetrin askeleissa ja kuvien päällekkäisyys on vähintään 60 prosenttia. Liikkeen aikana rata saa kaareutua maksimissaan 18 astetta. Yhden askeleen vaatima laskenta-aika vaihtelee kahdesta kolmeen minuuttia visuaalisen odometrian ollessa käytössä.

Marsin pinta asettaa toisinaan myös sellaisia haasteita, joissa tämä menetelmä ei tuota hyväksyttävää tulosta. Laajat hiekka-aavikot saattavat sisältää liian vähän konkreettisia piirteitä, jotta niitä olisi tarpeeksi algoritmia varten. Tämän lisäksi edellä mainittujen optimaalisten liikeratojen rikkominen tuotti algoritmille hankaluuksia. Vääriä, mutta mönkijän visuaalisen odometrian oikeaksi mieltämiä ratkaisuja syntyi kahden ajon aikana alkuperäisen konfiguraation takia, jossa kuvista löydettyjen piirteiden ei tarvinnut kattaa riittävän isoa aluetta. Muuttamalla vaadittavasta kattavuudesta kertovaa parametria ongelma korjattiin. Ainoa jatkuvaan tarkkailuun asetettu ongelmakohta ilmeni Opportunity 235:en päivän kohdalla, kun liian piirteetön maasto keskitti visuaalisen odometrian huomion mönkijän omaan varjoon. Tämä tuotti täysin virheellisen tuloksen arvioon kuljetusta matkasta ja tapahtuman jälkeen ohjaajien on täytynyt ottaa mönkijöiden varjot huomioon antaessaan niille käskyn nojautua visuaaliseen odometriaan liikkumisen aikana.

Visuaalisen odometrian lisäämä autonomisen liikkumisen turvallisuus taas oli yksi selkeistä hyödyistä, joita mönkijöiden päivittäisessä operoinnissa havaittiin. Tämän lisäksi tarkempi liike mahdollisti kehittyneemmän kyvyn tehdä tieteellisiä havaintoja hankalassa maastossa esimerkiksi siten ettei ohjaajien tarvinnut varmistaa mönkijään kiinnitettyjen instrumenttien kohdentamista hitaasti välittyvän näköhavainnon perusteella.

Marsissa viettämästään 394:stä vuorokaudesta Opportunity nojautui

visuaaliseen odometriaan 75:nä päivänä. Ajopäiviä mönkijälle kertyi 172, joten algoritmia oli käytössä lähes joka toisena päivänä. Spiritin vastaavat luvut olivat 414 Marsin vuorokautta, joista ajopäiviä oli 184 ja visuaalista odometriaa käytettiin 52:na päivänä. Yhteensä kuvapareja analysoitiin tuona aikana 1 654 kappaletta, joista 1 418 pystyi arvioimaan mönkijän sijainnin sekä asennon riittävän tarkasti. [CMM05]

Huolimatta yllä mainituista ongelmista visuaalinen odometria on yksi mönkijöiden tärkeimmistä työkaluista turvallisen ja itsenäisen liikkumisen toteuttamisessa Marsin pinnalla. Havaituista hyödyistä kertoo myös se, että algoritmi on käytössä vuonna 2011 Marsiin lähetetyssä Curiosity-mönkijässä [GCV+12].

Lähteet

[CMM05] Yang Cheng, Mark Maimone, and Larry Matthies. Visual odometry on the mars exploration rovers. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 1, pages 903–910. IEEE, 2005.

[FS07] Dave Ferguson and Anthony Stentz. Field d*: An interpolation-based path planner and replanner. In *Robotics Research*, pages 239–253. Springer, 2007.

[GCV+12] John P Grotzinger, Joy Crisp, Ashwin R Vasavada, Robert C Anderson, Charles J Baker, Robert Barry, David F Blake, Pamela Conrad, Kenneth S Edgett, Bobak Ferdowski, et al. Mars science laboratory mission and science investigation. *Space Science Reviews*, 170(1-4):5–56, 2012.

[Hon10] Wenxian Hong. *A study of fast, robust stereo-matching algorithms.* PhD thesis, Massachusetts Institute of Technology, 2010.

[KL02] Sven Koenig and Maxim Likhachev. D* lite. In *AAAI/IAAI*, pages 476–483, 2002.

[Mor80] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, DTIC Document, 1980.

[MS87] Larry Matthies and Steven A Shafer. Error modeling in stereo navigation. *Robotics and Automation, IEEE Journal of*, 3(3):239–248, 1987.

[Ste95] Anthony Stentz. The focussed dˆ* algorithm for real-time replanning. In *IJCAI*, volume 95, pages 1652–1659, 1995.