KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Star Tracker Algorithms and a Low-Cost Attitude Determination and Control System for Space Missions

Tjorven Delabie

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

January 2016

Star Tracker Algorithms and a Low-Cost Attitude Determination and Control System for Space Missions

Tjorven DELABIE

Examination committee:
Prof. dr. ir. Paul van Houtte, chair
(*preliminary defence*)
Prof. dr. ir. Joos Vandewalle, chair
(*public defence*)
Prof. dr. ir. Joris De Schutter, supervisor
Dr. Bart Vandenbussche, co-supervisor
Prof. dr. ir. Dirk Vandepitte
Prof. dr. ir. Tinne De Laet
Prof. dr. ir. Gaëtan Kerschen
(University of Liège)
Ir. Steeve Kowaltschek
(European Space Agency)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

January 2016

Uitgegeven in eigen beheer, Tjorven Delabie, Celestijnenlaan 300B box 2402, B-3001 Heverlee (Belgium)

ISBN XXX-XX-XXXX-XXX-X

D/XXXX/XXXX/XX

Preface

When you write a PhD book, like the one you are holding right now, you put months of work in getting your text just right. You want to explain everything thoroughly yet concise, you try to make the graphs self-explanatory, iterate on the conclusions, etc. And you do all this work, in the safe knowledge that hardly anyone who gets the book, will actually read all those technical chapters. Most people will not even go through these pages for more than 15 minutes. For all the hard work you did on all that scientific content, you already know that the one section that will actually speak to a larger audience, is this one right here. And while it's easy to feel frustrated about that fact, I'd just like to be pragmatic about it:

I would like to thank you Chapter 1. Oh Chapter 1, the way you introduce the interesting topics of this PhD to people amazes me every day. And you Chapter 2, your contents brought me to a conference in Hawaii, so how could I ever thank you enough? The Gaussian Grid algorithm you introduce brings a sublime mix of speed and accuracy to a field in which you previously had to choose one of the two. Chapter 3, it all started with you didn't it? Thomas and I started working on you during our thesis. Little did we know you would grow up to discuss one of the most robust lost-in-space algorithms ever created! This word of thanks would be very incomplete without you Chapter 4, boldest of them all. As a true non-conformer, you never shy away from bold claims like: "The world is flat". You present the fastest tracking algorithm ever created, how about that! In every group, there is always that one guy that's pushing the limits. I'm talking about you Chapter 5, you who take the robustness and speed of AIM that one step further! Chapter 6, the one who stands out! In a book with a very concentrated focus on star tracker algorithms, you just seem to be able to handle everything. The ADCS for CubeSats that you present will bring great agility and accuracy to small satellites. Never stop pushing technology forward, Chapter 6. And last but not least, thank you Conclusion, for giving

the hasty reader of this book a place to go to. From the bottom of my heart, I hope you guys never change! Because that would most likely mean you contain errors...

Een doctoraat is nooit het resultaat van één iemand alleen. Ik wil in de eerste plaats Joris en Bart bedanken om mijn promotoren te zijn. Joris, bedankt om mij de vrijheid te geven om mijn onderzoek en werk zelfstandig in te plannen. Voor iemand die houdt van flexibiliteit is dat veel waard! Tijdens mijn onderzoek op de invarianten-methode en tijdens meetings met thesis studenten was ik altijd onder de indruk van je wil om iets helemaal tot in detail te begrijpen. Dat is iets waar ik, als ongeduldige mens, veel van kan opsteken. Bart, de allereerste keer dat we samen zaten, waren we aan het discussiëren over hoe hoog de zelfgemaakte raket van Jeroen gelanceerd zou kunnen worden. Jouw gok was: "Als hij van de polders lanceert, -4m". En toen dacht ik dat het wel goed zou komen. Bedankt voor de vele informatie rond sterren, optica en sensoren, voor de samenwerking rond SIMBA, ITTs en andere projecten en voor de grappige momenten. Je scepsis rond het baffle project wordt je graag vergeven ;-).

Ik wil Dirk en Tinne, de assessoren van dit werk ook bedanken voor de vragen en discussies. Dirk, bedankt voor de samenwerking op het SIMBA project. Je technische inbreng en gevoel voor diplomatie zijn hier al erg nuttig geweest. Maar het meest onder de indruk ben ik nog altijd van het feit dat je tijdens de lessen Mechanica uit de losse pols een betere cirkel op het bord kon tekenen dan ik met mijn passer. Tinne, ik ben erg blij dat je lid bent van mijn doctoraatsjury. Ik heb samen met jou meetings gehad over uiteenlopende onderwerpen: Invarianten, state estimation, star trackers, etc. en ik ben altijd onder de indruk over de snelheid waarmee je nieuwe concepten snapt en doorgrondt.

Gaëtan, I would like to thank you for being one of my external jury members. Your expertise on small spacecraft has led to interesting questions and remarks. I would like to thank you for the short time we worked together within the QB50 project. When the B3LSat was not selected and our CubeSat team was separated by Belspo, my main regret was that the collaboration of KUL with you and Lamberto stopped, because I had really enjoyed it. Steeve, I was very happy that you wanted to be a part of my PhD jury. As one of the more "industrial side" experts on star trackers, I believe you have made this work more relevant for industry. I would also like to thank you for the fun moments on the different conferences!

Tijdens mijn doctoraat kreeg ik hulp van een heel aantal mensen. Bedankt Gert voor de goede raad, hulp bij testopstellingen en all-round sympathieke persoonlijkheid. Ik wil zeker ook Bram bedanken om in het SIMBA project de

software/elektronica kant grotendeels op zich te nemen! Mijn dank aan Johan, Eddy en Gunther die in de mechanische werkplaatsen onderdelen gemaakt hebben. Ook de elektronische dienst op Mechanica, Jean-Pierre en Bertram, en op Fysica wil ik bedanken om altijd meteen klaar te staan om mij te helpen met kabeltjes en elektronische componenten. Lieve, Valérie, Marina, Stephanie, Karin, Anja, Regine, Marijke en Carine, onze bakens van organiseervermogen, administratieve vaardigheid en simpelweg charme bij Mechanica verdienen zeker een vermelding.

Of course there are a lot of colleagues that make work more fun! Antonio who taught me Italian swear words, Onur with his great sense of humor and slightly less great AOE-skills. Frederik en Christophe worden bedankt om de edele West-Vlaamse taal te laten weerklinken binnen ons bureau. Matt for the fun talks and work with Diego. Simona, my apologies for mistakenly unplugging your pc while you were doing a big simulation. Sergio, for keeping your cool under all circumstances ;-). Caspar, for supplying me with stories so I could tell people: "you will never believe what this guy in my office did this weekend." Sikandar, because without his leadership and general awesomeness, this PhD would never have been completed. Niels, for the nice office jokes and good cheese. Stefan, for taking office party initiatives. Iris, Alex and Iratxo, for bringing Spanish charm into the office. Although we should be honest, it's mainly Iris that brings the charm. But you guys are cool too ;-).

Talking about Spanish charm brings me immediately to Ini! I would like to thank you for making the conference in Portoroz and a couple of my trips to ESTEC a lot more fun! Since you are "the good friend" I am sure that you will pay me a visit now and then when you live in the south of France!

Ik wil zeker ook mijn thesisstudenten bedanken die in meer of mindere mate hun bijdrage geleverd hebben tot ons ruimtevaartonderzoek. Johan, Martijn, Rutger, Ben, Diego, Dirk, Francois-Xavier, Ingmar, Jeroen, Milana en Wim, bedankt voor de leuke samenwerking!

Daarnaast zijn er nog een heel aantal mensen die niet echt bijgedragen[1] hebben aan dit doctoraat, maar desalniettemin een vermelding krijgen!

Maarten! Merci voor de lunches, te zeldzame gezelschapsspelnamiddagen, AOE-sessies, grappige gesprekken en om mij kennis te laten maken met Epic Sax Guy! Kathleen, bedankt om er voor te zorgen dat Maarten niet elke dag met een CW t-shirt rondloopt. Wally wil ik bedanken voor de lunches met schellekes

[1] Vrienden, waar zaten jullie eigenlijk toen ik vast zat op bepaalde problemen, de fout in mijn code niet vond, etc.? Het wordt tijd dat jullie eens stevig aan introspectie doen...

cola, croques en Barbie. Stephanie, merci voor de blogverhalen tijdens jullie wereldreis! Altijd leuk om te lezen. Natuurlijk wil ik Maarten en Wally ook bedanken voor de leuke samenwerking in SpaceBillboard. “Gepubliceerd raken in Dag Allemaal” is bij deze ook alweer van onze bucket list geschrapt. Piet et Caroline worden bedankt voor hun eeuwig enthousiasme! En allen samen voor de instant-traditie van kasteelschietingen! Aurélie en Katrien, voor de Lodge-lunches met filosofische gesprekken!

Stijn wil ik bedanken voor zijn ongeëvenaard enthousiasme als het op AOE aankomt, voor zijn basiskennis West-Vlaams en om “Castle On A Cloud” van“Les Miserables” te herkennen toen ik het floot. Kortom wil ik Stijn bedanken voor zijn kennis van hoogstaande cultuur. Ik ben Thomas dankbaar voor een heel leuke thesis, de beste hamburgers die ik ooit gegeten heb, leuke avonden en etentjes, galleons en fireships en goede humor! Lotte en Gwenny wil ik bedanken om in een bende burgiemannen een baken van charisma en schoonheid te zijn. Al is het nu toch al erg lang wachten op dat spraypaint pilotenpak, Lotte...

Voor een leuke studententijd vol winter-BBQs, grillsessies, cafébezoeken en colliepinne-avonden bedank ik Collie, Ciska en Julie[2]. Het is fantastisch om jullie ook nog in post-studententijden vaak te zien! Tijdens mijn studententijd had ik ook de meest sexy kotgenoten die een mens zich kan inbeelden: Lynn, Elise, Charlie en, ik geef toe, Maxime. Ook semi-kotgenote en dans/verjaardagsliedjes/gitaarwonder Heleen kan niet ontbreken!

Voor de fijne mannenweekendjes, uitgangsavonden, rodedraad spelletjes, crimiclown-sessies en andere leuke avonden bedank ik natuurlijk eerst en vooral Pieter Vandervelden. Dat hoort zo. En dit is, laat ons eerlijk zijn, ook het enige echt perfect gepaste moment om deze bedanking te uiten. Merci VDV voor het enthousiasme en de creativiteit in drankspelletjes. Bedankt (arro-)Piet voor legendarische uitspraken als daar zijn: “t’è woar é” en voor de goocheltrucs! Mijn dank aan Bman voor de cultureel hoogstaande opmerkingen en de fantastische metaforen met Usain Bolt. Merci Wouter om de rust in deze groep te helpen brengen! Josn wordt bedankt omdat hij een motiverend licht in de duisternis is, want “ajje da wilt, ton kundje dadde”. Vanzelfsprekend bedank ik stijlicoon en jeneverexpert Collie voor alle leuke momenten.

Verder zorgen de al-dan-niet spontane avonden met Celine, Gert, Pieter, Pauline, Katrien, Dieter, Tim en Griet ook altijd voor ontspanning! Net zoals de leuke meetings met birthdaybuddy Ronnie en topadvocaat Klaas! Bij deze wil ik ook nog wetenschappelijk noteren dat Heule een dorp is waar je door rijdt. Het

[2]Ik laat PJ er even uit omdat ervaring leert dat hij tijdens de Laudatio sowieso nog uitgebreid aan bod zal komen.

staat nu in een doctoraat dus het is waar. Bram, An, Koen, Leen, Ileen en Bram, bedankt voor de fijne namiddagen met brave kindjes en bedankt voor de interesse in mijn doctoraat en SpaceBillboard!

Ik wil bij deze ook al mijn medevrijwilligers van het weeshuis voor puppy'tjes bedanken!

Ah, ik moet vanzelfsprekend ook nog Daan Glas bedanken. Stel je voor dat ik het zou vergeten, genant. Merci Glas voor de Baarr-avonden, weekendjes, organisatievermogen, South Park gesprekken en AOE-events!

En dan zijn er natuurlijk nog de stuk voor stuk fantastische vrienden van dichter bij huis! Samen zijn met jullie zorgt altijd voor sjieke momenten: De Molecule Vedettenparade, oudejaarvieringen, iets gaan drinken op café, optredens, iets met cavia's (we weten nog niet wat precies, maar iets met cavia's) en weekendjes met PolyPuly en MuchioMulli! Ik kan niet anders dan heel gelukkig zijn met vrienden als Jan, met wie ik enkele van de coolste reizen ooit gemaakt heb en van wie de creativiteit mij altijd verwondert, Griet, zonnetje,Willy Sommers-polonaise partner, Jelle, stud die liedjes aan mij opdraagt :-), Sofie, van wie ik het gevoel voor humor erg kan appreciëren, Duff, van wie ik het gevoel voor humor soms kan appreciëren ;-), Tine, die ondanks de jongste te zijn, mij altijd de indruk geeft de meest wijze te zijn, Bertje voor zijn ongeëvenaard enthousiasme en Excellent organiseervermogen, Charlotte, om haar eigen immer stijlvolle zelf te zijn, Velghe, die meer energie heeft dan om het even welk ander persoon die ik ken, Julie, die daar toch wel heel goed mee om kan en altijd cool blijft, Steven, die ik bij deze helemaal niet bedank omdat hij maar niet altijd moet lachen met mijn grotere handen, Robbe, die één van de coolste lachen heeft die ik ken, Janneke, die ons waarschijnlijk binnenkort collectief zal bijstaan met wijze babyraad, Pieter, die altijd fantastisch is op mannenweekend, Eline, om bij deze meteen ook goed te maken dat ik niet op je gestemd heb, Bram, omdat hij de beste is die Eline ooit al meegebracht heeft, Edith, die ondanks haar "fellen teut" toch een braaf kindje is, Gabòr, mert ö egy jó srác nagy mell (hopelijk klopt dat), Lisa, die mij een reistalisman gegeven heeft die ik nog altijd in mijn rugzak heb zitten en afsluiten doe ik met Carlo, "omdat het goed geweest is zeker gastjes?" En uiteraard voor de Game of Thrones gesprekken en de afspraken in Brussel en Leuven.

Ik wil bij deze ook enkele familieleden bedanken: PeAndré, MéAgnes, Christ, Monica, Tantan, Levy, Julie, Sharon en dan natuurlijk buiten de inner-circle, Patrice ;-). En ook de schoonfamilie: Jan, Annemie, Maarten, Chris, Sabine en zelfs Tine wil ik bedanken voor de goede ontvangst!

Ik ben Franky dankbaar voor het oprichten van mijn fanclub en om mij de ware schoonheid van Pugs te leren zien. Merci aan Gerard om mij te vergezellen naar Hawaii! Bij deze wil ik ook gebruik maken van de gelegenheid om voor eens en altijd te definiëren dat de voorkant van het perron bepaald wordt door de richting waarin de trein zal wegrijden. Laura, kan jij het hem anders nog eens uitleggen?

Ongeveer een jaar geleden zei iemand tegen mij: “Ik denk dat jij een goede kindertijd gehad hebt.” Waarom hij dat zei is mij een absoluut raadsel, maar ik weet wel dat ik zonder twijfelen dacht dat hij gelijk had. Mama en Pitoe, ik zie rondom mij weinig ouders die zo zorgzaam en liefdevol zijn en ik prijs mezelf dan ook erg gelukkig. Door nieuw-gedefinieerde landenkaarten, mansions en nutty situations zou ik ons gezin geen oase van rust noemen, maar er is altijd iets te beleven en het is altijd heel leuk om thuis te komen.

Sofietje, ik kan je voor heel veel dingen bedanken. Voor het liefste glimlachje ter wereld, het feit dat je mijn meest flauwe mopjes de beste vindt, je kleine verhaaltjes, dat je mij altijd staat uit te zwaaien als ik weg ga, je lieve briefjes en zoveel meer. Bedankt om er altijd voor mij te zijn en om je eigen lieve zelf te zijn. Ik kijk er elke dag opnieuw naar uit om samen met jou door het leven te stappen.

Abstract

The attitude determination and control system determines and controls the orientation of the spacecraft. This system is crucial in the majority of space missions to e.g. point a camera to a star or direct an antenna to a ground station. Increasingly complex missions drive the need for higher accuracy, while the growing number of small spacecraft requires high robustness and low computational cost. This work focusses on the star tracker, a sensor that takes an image of the stars and compares it to a database with known star positions to determine the spacecraft attitude. Algorithms are developed with high accuracy, high robustness and low computational cost. A full attitude determination and control system for a class of nanosatellites, called CubeSats, is also presented.

The centroiding algorithm determines the centroid of stars in the camera image. The developed algorithm uses closed form expressions to fit a model to the measured star data. The use of model fitting leads to high accuracy, while the closed form expressions keep the computational cost low. The accuracy is in the range of the most accurate algorithms and the computational cost is in the range of the fastest algorithms.

The lost-in-space algorithm matches the stars in the camera image to stars in the database. The developed algorithm, which depends on the Shortest Distance transform is very robust to false stars, distortions in the image and missing stars. On top of that, it offers a reliable quality value for the result, which further increases the robustness.

The tracking algorithm finds the transformation values between camera and database stars and determines the attitude based on these values. The AIM algorithm developed in this work is the fastest tracking algorithm available. Its novel approach to solving the tracking problem can be exploited to increase the robustness and decrease the computational cost.

The algorithms deliver similar accuracy as the most accurate state of the art algorithms. Their computational cost is lower and they have higher robustness. These algorithms and the reaction wheels designed at KU Leuven are enablers for a high accuracy attitude determination and control system for CubeSats.

A low-cost high accuracy attitude determination and control system for CubeSats is under development at KU Leuven. The system delivers high pointing accuracy for low volume, weight and power consumption. It opens up potential for small satellite missions that have higher demands on the pointing accuracy. The system is introduced and its performance is analysed in simulations.

Beknopte samenvatting

Het standbepalings- en controlesysteem bepaalt en controleert de oriëntatie van een satelliet. Dit systeem is cruciaal in de meeste ruimtemissies om bv. een camera naar een ster te richten of een antenne naar een grondstation te richten. Met de toenemende complexiteit van missies neemt de vereiste nauwkeurigheid toe. Een groeiend aantal kleine satellieten heeft nood aan hoge robuustheid en hoge rekenefficiëntie. Dit werk focust op de sterrensensor die een beeld van de sterren neemt en dit vergelijkt met een cataloog van gekende sterposities om de stand te bepalen. Algoritmes met hoge nauwkeurigheid, hoge robuustheid en hoge rekenefficiëntie werden ontwikkeld. Een standbepalings- en controlesysteem voor een klasse van nanosatellieten, CubeSats genaamd, wordt voorgesteld.

Het centrering-algoritme bepaalt het middelpunt van de sterren in het beeld. Het ontwikkelde algoritme gebruikt analytische formules om een model door de gemeten sterdata te fitten. Het gebruik van een gefit model brengt hoge nauwkeurigheid met zich mee, terwijl de analytische formules de benodigde rekenkracht laag houden. De nauwkeurigheid is gelijkaardig aan die van de meest nauwkeurige bestaande algoritmes en de rekenefficiëntie is ongeveer gelijk aan die van de snelste algoritmes.

Het verloren-in-de-ruimte-algoritme combineert de sterren in het camerabeeld met sterren uit de database. Het ontwikkelde algoritme gebruikt de Kortste Afstand transformatie en is zeer robuust tegen valse sterren, storing in het beeld en niet-gedetecteerde sterren. Daar bovenop levert het een betrouwbare kwaliteitswaarde die de robuustheid verder verhoogt.

Het volgalgoritme bepaalt de stand van de satelliet op basis van de transformatiewaarden tussen camera en databasesterren. Het AIM-algoritme dat ontwikkeld werd is het snelste volgalgoritme vergeleken met andere algoritmes in de literatuur. De nieuwe aanpak om het volgprobleem op te lossen laat toe

de robuustheid en rekenefficiëntie verder te verhogen.

De algoritmes leveren gelijkaardige nauwkeurigheid als de bestaande algoritmes. De rekenefficiëntie en robuustheid zijn hoger. Deze algoritmes en de reactiewielen ontworpen aan de KU Leuven laten toe een standbepalings- en controlesysteem met hoge nauwkeurigheid te ontwikkelen voor CubeSats.

Een lage-kost standbepalings- en controlesysteem met hoge nauwkeurigheid voor CubeSats wordt momenteel ontwikkeld aan de KU Leuven. Het systeem beoogt hoge richtnauwkeurigheid voor een laag volume, gewicht en vermogenverbruik. Het opent mogelijkheden voor missies met kleine satellieten die hogere vereisten hebben op het vlak van richtnauwkeurigheid. Dit systeem wordt geïntroduceerd en de prestaties wordt geanalyseerd.

Abbreviations

ADCS	Attitude Determination and Control System
AIM	Attitude estimation using Image Matching
ARW	Angular Random Walk
COG	Center Of Gravity
DC	Dark Current (noise)
DC motor	Direct Current Motor
DT	Distance Transformation
ESA	European Space Agency
ESOQ	EStimator of the Optimal Quaternion
ESOQ2	EStimator of the Optimal Quaternion 2
Flops	Floating-point operations
FOAM	Fast Optimal Attitude Matrix
FOV	Field Of View
FWC	Full Well Capacity
FWHM	Full Width at Half Maximum
GG	Gaussian Grid
HCOG	Hybrid Center Of Gravity
HGG	Hybrid Gaussian Grid
IARU	International Amateur Radio Union
IGRF	International Geomagnetic Reference Field
IWCOG	Iteratively Weighted Center Of Gravity
IWT	Agency for Innovation by Science and Technology in Flanders

KUL	KU Leuven
LSQ1	Least Squares 1 Dimensional with Reduced number of rows
LSQ1	Least Squares 1 Dimensional
LSQ2	Least Squares 2 Dimensional
MEMS	Micro Electro-Mechanical Systems
MTM	Magnetometer
MTQ	Magnetorquer
NASA	National Aeronautics and Space Administration
NEA	Noise Equivalent Angle
P-POD	Poly-PicoSatellite Orbital Deployer
PCB	Printed Circuit Board
PSF	Point Spread Function
QUEST	QUaternion ESTimator
RMIB	Royal Meteorological Institute of Belgium
RMS	Root Mean Square
RRW	Rate Random Walk
RW	Reaction Wheel
SGP-4	Simplified perturbations model
SIMBA	Sun-earth IMBAlance
SVD	Singular Value Decomposition
TLE	Two-line Element Set
TRL	Technology Readiness Level
WCOG	Weighted Center Of Gravity

List of Symbols

N_p	The number of database images used in the lost in space algorithm
N_{tot}	The minimum of the number of stars in the camera image and the database image
O	Percentage overlap between two images
k	The number of elements selected from the set with size n
n	The size of a set of elements
n_{comb}	A number of combinations
n_{perm}	A number of permutations
α_b	Right ascension boundary angle for the database image
δ	Declination angle
γ	The field of view
σ	Standard deviation
Ω	A set of points used to calculate a distance transformation
Ω^c	A subset of points used to calculate a distance transformation that lies within Ω
σ_{pos}	The noise on the star centroid position
$(\hat{i}, \hat{j}, \hat{k})$	The unit vector coordinates of a database star
$(\hat{i}_r, \hat{j}_r, \hat{k}_r)$	The unit vector coordinates of a database star, after rotation by the inverse of q_{dat}

$(\hat{x}_i, \hat{y}_i)$	The coordinates of database star i in the focal plane
$(\hat{x}_{it}, \hat{y}_{it})$	The coordinates of the transformed database star i in the focal plane
(b, l)	Maximum values for the x and y coordinates of the focal plane
(i, j, k)	The unit vector coordinates of an observed star
(x_0, y_0)	The intersection of the focal plane and the optical axis
(x_b, y_b)	The centroid of a Gaussian
(x_e, y_e)	The estimated star centroid
(x_i, y_i)	The coordinates of observed star i in the focal plane
(x_t, y_t)	The true star centroid
α_r	The angle in arc seconds over which the coarse estimate, q_{dat}, is rotated away from the true attitude over three axes
β	The vector with the parameters to be determined in the least squares approach
$\Delta_{cp,cg}$	The distance between the center of gravity and the center of pressure
$\epsilon_{x,y}$	Value of the measurement noise
Γ_{AIM}	Cost function in the derivation of the AIM algorithm
Γ_{GG}	Cost function in the derivation of the Gaussian Grid algorithm
λ_0	The sum of the weights given to the star pairs in the cost function
λ_{max}	Maximum eigenvalue calculated in the QUEST attitude calculation
$\mathbf{b}_i$	A unit vector of star i, observed in the spacecraft body frame
$\mathbf{b}_r$	The unit vector of removed star r, observed in the spacecraft body frame
$\mathbf{r}_i$	A unit vector of star i, observed in a reference frame

$\mathbf{r}_r$	The unit vector of removed star r, observed in a reference frame
$\mathbf{r}_{it}$	The unit vector of star i, after rotation by the estimated attitude quaternion
μ	The standard gravitational parameter of Earth
ω_{rw}	The rotational rate of a reaction wheel flywheel
ϕ	Roll angle of the spacecraft attitude
ψ	Yaw angle of the spacecraft attitude
ρ	The atmospheric density
σ_x	x-Standard deviation of a Gaussian
σ_y	y-Standard deviation of a Gaussian
σ_{ro}	Standard deviation of the read out noise
$\sigma_{sn,i,j}$	Standard deviation of shot noise at pixel with coordinates (i, j)
θ	Pitch angle of the spacecraft attitude
θ_{dM}	Maximum allowed angle between a camera star vector and its corresponding database star vector
θ_d	The angle between a camera star vector and its corresponding database star vector
θ_e	The angle in arc seconds between the true quaternion - q_t - and the estimated quaterion - q_e
θ_{or}	The orientation angle of the CubeSat
A	Orthogonal attitude matrix which minimizes the Wahba cost function.
a	Maximum of a Gaussian function
A_s	Maximal surface of the CubeSat
A_f	The frontal surface of the CubeSat
b	The difference between the maximum and minimum focal plane coordinate in the x-direction

c	The speed of light
C_d	The drag coefficient of the CubeSat
d	The distance squared used in the Gaussian Grid analysis
d_i	The distance squared between camera image star i, and its corresponding transformed database star in the focal plane
e	The error between the measured value and the value of the model Gaussian
E_{cent}	The average centroiding accuracy
F	The focal length of the optical system
F_s	The solar constant
h_{rw}	The momentum of a reaction wheel
I_x	The moment of inertia around the x-axis
I_y	The moment of inertia around the y-axis
I_z	The moment of inertia around the z-axis
I_{ij}	Intensity at pixel with coordinates (i, j)
I_{rw}	The moment of inertia of a reaction wheel flywheel
k_i, h_i	Parameters of the polynomials to transform the measured coordinates to corrected coordinates for the camera distortions
k_{iin}, h_{iin}	Parameters of the polynomials to transform the corrected coordinates for camera distortions to measured coordinates
l	The difference between the maximum and minimum focal plane coordinate in the y-direction
M	The measured value of the signal
M_{mtq}	The magnetic moment generated by the magnetorquer
N_e	The total number of electrons captured by the detector
n_o	The number of outlier stars in the data
n_s	Number of stars used in the attitude estimation algorithm

N_{bits}	The number of bits used to represent the pixel intensity
n_{pix}	The number of pixels in one row of the image
n_{pr}	The number of pixels in a row or column of the pixel cluster
NEA_{cross}	The cross boresight Noise Equivalent Angle
P	Covariance of the estimation error
P_e	The factor with which the variance of the position error was multiplied for outliers
P_o	The factor with which the standard deviation on the error of the outlier stars is higher than that of the other stars
q_{dat}	The quaternion representing the attitude at which the database image is selected
q_{diff}	The quaternion representing the estimated difference in attitude between q_t and q_{dat}
q_e	The quaternion representing the estimated attitude of the satellite
q_t	The quaternion representing the true attitude of the spacecraft
t_x	Distance over which the database stars are translated in the x-direction
t_y	Distance over which the database stars are translated in the y-direction
T_{ad}	The atmospheric drag disturbance torque
T_{cyc}	The total cyclic disturbance torque
T_{gg}	The gravity gradient disturbance torque
T_{mf}	The magnetic field disturbance torque
T_{mtq}	The torque generated by the magnetorquer
T_{sec}	The total secular disturbance torque
T_{sp}	The solar pressure disturbance torque
T_{tot}	The total disturbance torque

V_{mx_i}	Summed intensity of the x-marginal pixel values
$V_{x,y}$	The measured pixel value at coordinates (x, y)
Vr_{ij}	Rounded value of the intensity at pixel with coordinates (i, j)
w_i	Non-negative weight of star i
$w_{i,j}$	Weights used in the Gaussian Grid algorithm for pixel (i, j)
W_{ij}	Value of the weighting function at pixel with coordinates (i, j)
D	The dipole of the CubeSat
i	The incident angle of the Sun
M	The magnetic dipole moment of Earth
q	The reflection factor of the CubeSat
R	The radius of Earth
V	The velocity of the CubeSat

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This introduction positions the presented work within the space mission. The novel developments are discussed briefly and the overall structure of the dissertation is presented.

1.1 Space missions

Space missions are among the most complex human endeavors. Apart from political, economical and managerial challenges, they require the combination of a wide range of scientific disciplines. A successful space mission relies on the collaboration of an often multinational team of astronomical scientists, telecommunication engineers, mechanical engineers, etc.

Figure 1.1 is used to position the work that has been done in this PhD, within the larger scope of a space mission.

A spacecraft mission can be divided into three major parts:

The ground segment is the part of the mission that has its place on Earth. The mission control center, which interacts with, and controls the spacecraft once it is in space, is part of this ground segment. The ground support equipment that is used to qualify the spacecraft can also be attributed to the ground segment.

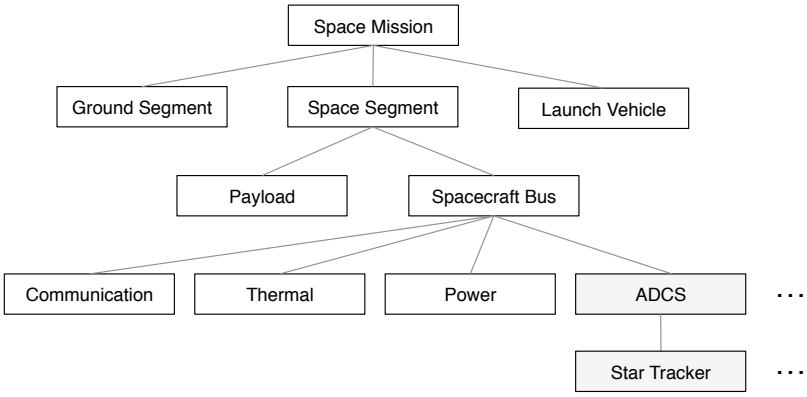


Figure 1.1: A schematic of the elements in a space mission.

The launch vehicle is the part of the mission that brings the spacecraft into space. Typically, a spacecraft is placed on board of a rocket and is launched using chemical propulsion. New methods to bring a spacecraft into orbit with a reusable space-plane [47] or even with an elevator [119] have been proposed.

The space segment is the part of the mission that will function in space. This is the part of the mission that will be our focus.

The three segments greatly influence each other. The conditions during launch (such as vibrations, acoustics, shock, etc.) have an important impact on the design of the spacecraft. The orbit in which the spacecraft is launched will determine how the ground station can communicate with the spacecraft, etc.

Within the space segment, we can further make a distinction between the payload and the spacecraft bus.

The payload is the "useful part" of the spacecraft. It is the reason why a spacecraft mission was developed. The payload could be an optical instrument to observe Earth or stars, a communication module to transmit TV-signals, a biological experiment, a technology one would like to demonstrate in space, etc.

The spacecraft bus comprises the components on board of the spacecraft that make sure the payload can perform well. The spacecraft bus powers the spacecraft, controls the temperature, communicates with the ground station, controls the orientation and position of the spacecraft, etc. This work has developed new techniques for one of the systems of the spacecraft bus, namely the Attitude Determination and Control System (ADCS).

1.2 Attitude Determination and Control

Many missions require control of the orientation - or attitude - of the spacecraft. For Earth observation missions, the payload needs to be pointed towards Earth to make measurements, communication satellites need to point their antennas to the ground stations and astronomical missions need to point their instruments to stars or other points of interest. The system that assures this control of the attitude of the spacecraft is called the attitude determination and control system.

To fully control the attitude of a spacecraft, a feedback loop as shown in Figure 1.2 is typically used. The attitude of the spacecraft is estimated in the "Determination" part. The estimated attitude and rotational rate are then compared to the desired attitude and rotational rate, the "Reference". Based on this reference, the attitude is controlled in the "Control" part. Unfortunately, there are also disturbances acting on the spacecraft, which change the attitude. Examples are the atmospheric drag, solar pressure and magnetic disturbance torques.

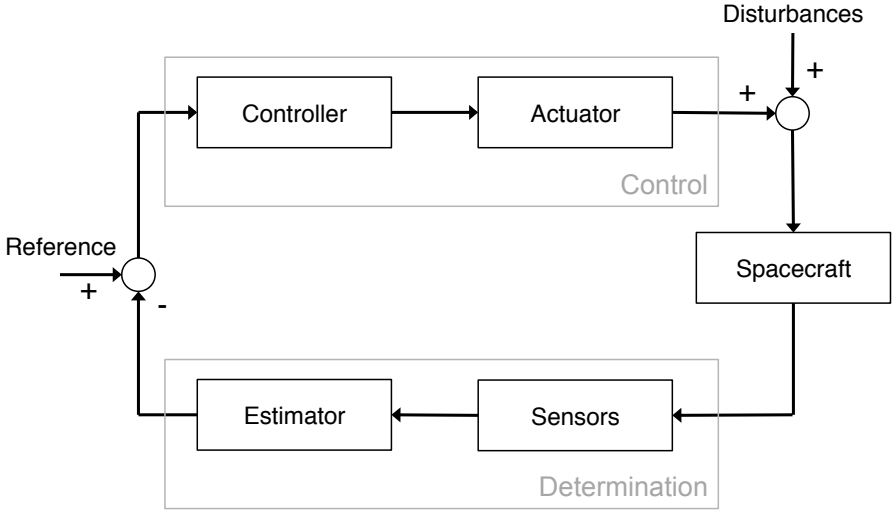


Figure 1.2: A schematic of an attitude determination and control system.

1.2.1 Attitude Determination

To control the attitude of a spacecraft to a desired attitude, knowledge of the current attitude of the spacecraft is needed. Several sensors to measure the attitude of a spacecraft exist. The most common ones are listed below [115]:

- **Sun Sensors** use photo-sensitive sensors to determine the direction of the Sun relative to the spacecraft. Together with the known position of the sun in the inertial frame, this gives information on the attitude.
- **Earth Sensors** use visual or infrared sensors to determine the position of the Earth relative to the spacecraft. Together with the known position of the Earth in the inertial frame, this gives information on the attitude.
- **Magnetometers** measure the direction of the magnetic field of the Earth. Together with models of this magnetic field, this gives information on the attitude.
- **Star Trackers** take an image of the stars and locate the stars within that image. Together with star databases [36] that accurately describe the position of the stars in an inertial frame, this sensor can determine the attitude of the spacecraft.
- **Inertial Measurement Units** measure the rotational rate of the spacecraft. As opposed to the previous sensors, they do not measure the absolute attitude but rather the change in attitude.

What the absolute attitude sensors have in common is that they measure a vector towards or along a physically measurable entity in the body frame and compare that to a known or modeled vector towards or along that same entity in an inertial frame. To fully determine the attitude, two vectors are needed [79]. The star tracker determines vectors towards multiple stars and can on its own determine the attitude. For the other absolute attitude sensors, at least two need to be combined to determine the attitude.

The output of these sensors is passed to the estimator. The estimator uses these measurements together with a model of the system and the uncertainties to estimate the attitude and rotational rate of the spacecraft.

1.2.2 Attitude Control

Based on the difference between the desired and current attitude (and rotational rate), the controller determines the action of the actuators. There are several types of actuators to control the attitude of a spacecraft. The most common are listed below [115]:

- **Magnetorquers** are solenoids that generate a magnetic field. This magnetic field interacts with the Earth magnetic field to create a torque.
- **Reaction Wheels** are controllable flywheels that spin up or down to create a torque. Because of conservation of angular momentum, when a reaction wheel spins up in one direction, the satellites spins up in the other.
- **Thrusters** generate a torque by firing at least two of them in opposite directions, each on another side of the rotational axis. This way there is no translation, but only a rotation.

1.3 Star Tracker

Narrowing down the scope, the focus of the majority of this work is on one of the sensors of the attitude determination and control system, namely the star tracker. This sensor is inherently the most accurate attitude determination sensor [74] and can determine the attitude with errors in the arc second or even sub-arc second range.

The main goal of this work is to present novel algorithms that improve star tracker performance. Before these algorithms are discussed, an overview of the algorithm cycle in a star tracker is given. After this, the main performance parameters of these star tracker algorithms are assessed.

1.3.1 Star Tracker Functionality

The procedure to go from a star image to knowledge of the attitude of the spacecraft is depicted in Figure 1.3. First, a camera on board of the spacecraft takes an image of the stars. This image can be filtered to remove noise, stray

light, hot pixels [41] or other elements that reduce the quality of the image. The stars are typically located in the image by identifying the areas with brighter pixels than the average background.

The image is then processed by the **centroiding** algorithm to find the centroids of the stars in the image as accurately as possible. State of the art algorithms can determine the centroid to sub-pixel accuracy. After this step, a distortion correction algorithm can be used to correct the disturbances introduced by imperfect optics or to compensate for the change in focal length with temperature of the star tracker.

For the stars in the image, the corresponding stars in the database are found next. If no previous knowledge of the attitude is available, this is done by the **lost-in-space** algorithm. This algorithm scans the entire star database and uses a pattern matching technique to link the stars in the image with the corresponding stars in the database. When there is a-priori knowledge, the search can be reduced around the previous attitude and a more simple **matching** algorithm can be used. The catalog positions are corrected for stellar aberration, which depends on the projected velocity of the spacecraft with respect to the tracking star direction. The diffraction in the star tracker optics will also depend on the color of the tracking star. The catalog position of the star is also corrected for this effect, depending on the expected location in the star tracker field of view.

With the coordinates of the stars known in both the camera image and the database image, the attitude can be determined using the **tracking** algorithm. This algorithm finds the transformation between camera and database star coordinates and this way determines the attitude of the spacecraft.

1.3.2 Main Performance Indicators

This work presents novel algorithms to improve the performance of the star tracker. To assess this improvement, it is first necessary to determine what constitutes a performance improvement. In other words, we will establish what the desired performance characteristics of a star tracker are. Three main parameters are identified.

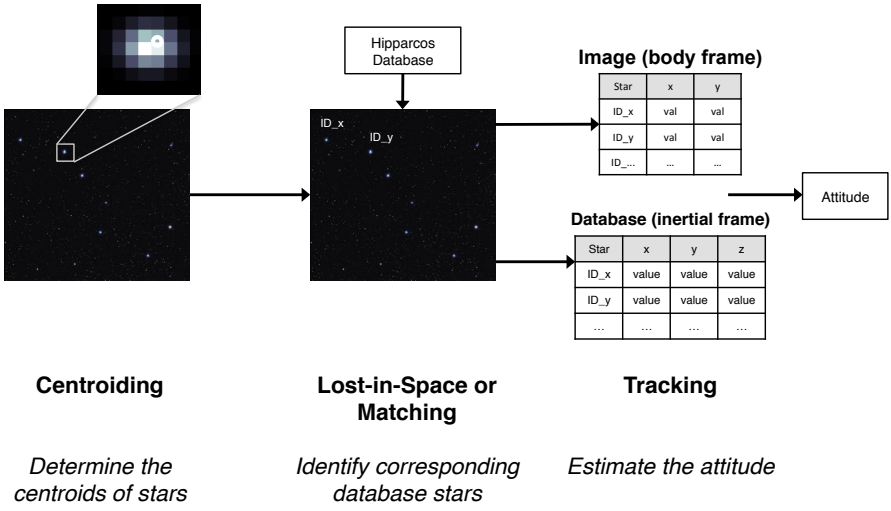


Figure 1.3: A schematic of the algorithm cycle of a star tracker.

Accuracy

When the star tracker gives a more accurate result, the pointing knowledge of the spacecraft improves. When the actuators allow it, this can also lead to an improved pointing accuracy. Improved pointing knowledge and accuracy have a positive effect on scientific output and can allow more complex missions requiring higher pointing accuracy to be flown.

Computational complexity

A reduction in computational complexity of the star tracker algorithms allows to get the result of the star tracker faster. This has a positive impact on the performance of the control loop and can again improve the pointing accuracy of the spacecraft. On the other hand, one can chose to use the reduction in complexity to implement processing components with less power. This can reduce the cost and power consumption of the mission. Additionally, the reduction in computational complexity can be used to implement more computationally complex and accurate algorithms in another stage of the star tracker. When the number of tracked stars is limited by CPU power, reduced algorithm complexity can allow us to track more stars in the star tracker algorithms, which improves the accuracy.

Robustness

A robust algorithm is an algorithm that continues to perform as required, even in the presence of disturbances. There are several examples of these disturbances for a star tracker. The detector could detect false stars coming from high-energy particles hitting the detector, light reflecting on an exhaust plume or in the case of the Rosetta mission, comet dust [37]. The detector could fail to detect a star because of dead pixels or the star position could be far off due to an uncalibrated distortion in the optics. Robustness to these distortions is a great advantage because it allows the spacecraft to benefit from the accurate star tracker measurements during a higher fraction of the orbit.

While these three characteristics matter for all algorithms, their relative importance is specific for each algorithm.

1.3.3 Performance Indicator Importance for each Algorithm

Centroiding

The centroiding algorithm is called for each star tracker measurement and for each star. Star trackers typically track around 10 stars. Therefore, the computational cost of this algorithm has a large impact on the total computational cost. The accuracy of the star tracker scales linearly with the centroiding accuracy. A robust centroiding algorithm could signal that a non-star signal was incorrectly identified as a star. A more robust algorithm is more desirable.

Lost-in-space

The lost-in-space algorithm is typically only called at the start of the star tracker procedure or when the attitude has been lost. During nominal operation of the star tracker, a faster and simpler matching algorithm can be used. While a faster algorithm is preferred, the computational cost of the lost-in-space algorithm is of less importance. The term accuracy is not really relevant, since the algorithm identifies the stars either correctly or not. The most important characteristic of this algorithm is its robustness. The star tracker procedure cannot be started without a correct initialization, given by the lost-in-space algorithm. Even in the presence of high disturbances (false stars, missing stars, optical distortions, etc.), this algorithm should still be able to identify the stars in the image.

Tracking

The tracking algorithm is called for each star tracker measurement and its computational cost therefore has an important impact. The accuracy has a direct impact on the accuracy of the star tracker. High robustness improves the availability and autonomy of the star tracker and is desirable.

1.4 Objectives

In the first part of this work, the development of a novel centroiding, lost-in-space and tracking algorithm is discussed. **The goal of this development is to increase the accuracy, robustness and computational efficiency of the star tracker by using novel algorithms.** Each chapter starts with an overview of the state of the art and makes an analysis of the disadvantages of current algorithms. The performance indicators, introduced earlier, are used in this analysis. The novel algorithm is then introduced and simulation tests are used to validate the performance of the novel algorithms and to compare it to the state of the art. The conclusion of every chapter emphasizes the improvements with respect to the performance indicators.

In the second part of this work, the development of a low-cost attitude determination and control system for CubeSats [117] is discussed. A CubeSat is a novel and small type of satellite which has great benefits for university projects. **The goal of this development is twofold: (1) The CubeSat ADCS will use the star tracker algorithms developed in this work and will allow to test them in space. (2) The ADCS will deliver high pointing accuracy and will be an interesting system in the growing field of CubeSats.** Furthermore, the CubeSat ADCS has for a large part been developed by KUL students. Developing this subsystem has granted them more insight in space missions and has given them hands-on experience with space technology.

1.5 Contributions

The star tracker algorithms that are developed in this work improve upon the state of the art when the overall performance in terms of accuracy, robustness

and computational efficiency are combined. The novel centroiding algorithm offers the same accuracy as the most accurate state of the art method for a significantly lower computational cost. It can also calculate the centroid an order of magnitude faster, be it then with slightly lower accuracy. The lost-in-space algorithm outperforms the state of the art in terms of robustness. The algorithm is extremely robust to false stars, missing stars and distortions in the image, while offering a very reliable criterion to assess the correctness of the result. The tracking algorithm that was developed is the fastest tracking algorithm available. It also has an efficient procedure to increase the robustness and decrease the computational cost. As a whole, these algorithms greatly improve computational efficiency and robustness, while maintaining the accuracy. This makes them very useful for use in small satellites and to improve the availability of the star tracker.

The second part of this work has a much broader scope. The development of the CubeSat Attitude Determination and Control System builds greatly on the results of Master thesis students. The major contribution in this project was to analyse the mission and its ADCS requirements, outline the functionality of the ADCS and coordinate the development of the system. A simulation environment was set up to analyse the performance of the system. The resulting ADCS should have a pointing accuracy that exceeds that of the state of the art systems, thanks to an accurate and available star tracker and thanks to the reaction wheels.

1.6 Outline

The structure of this work mainly follows the algorithm cycle of the star tracker as shown in Figure 1.3. The chapter describing the development of an ADCS for CubeSats, the secondary goal of this work, is found at the end.

Chapter 2 reviews the state of the art in centroiding algorithms and presents a new algorithm: The Gaussian Grid algorithm. This algorithm uses closed-form expressions to calculate the centroid of a star with high accuracy and low computational cost.
Chapter 3 presents a lost-in-space algorithm based on the shortest distance transform. This algorithm has very high robustness to false stars, missing stars and distortions in star positions. Furthermore it gives a reliable confidence value to validate the result.
Chapter 4 discusses the tracking algorithm, called AIM. As opposed to the state of the art algorithms, it uses 2D-coordinates to solve the tracking problem.

This leads to a decrease in computational cost, while the accuracy can be maintained.
Chapter 5 elaborates the AIM algorithm. It discusses methods to improve the robustness and reduce the computational cost of the AIM algorithm.
Chapter 6 discusses the development of the attitude determination and control system for CubeSats. It starts with an introduction to the field of small satellites and gives a technical overview of the different systems of the KUL ADCS and its simulated performance.

Chapter 2

An Accurate and Efficient Gaussian Fit Centroiding Algorithm for Star Trackers

Delabie, T., De Schutter, J., and Vandenbussche, B. (2015) An Accurate and Efficient Gaussian Fit Centroiding Algorithm for Star Trackers. *The Journal of the Astronautical Sciences*, 61 (1), 60-84.

This chapter presents a novel centroiding algorithm for star trackers. The proposed algorithm, which is referred to as the Gaussian Grid algorithm, fits an elliptical Gaussian function to the measured pixel data and derives explicit expressions to determine the centroids of the stars. In tests, the algorithm proved to yield accuracy comparable to that of the most accurate existing algorithms, while being significantly less computationally intensive. Hence, the Gaussian Grid algorithm can deliver high centroiding accuracy to spacecraft with limited computational power. Furthermore, a hybrid algorithm is proposed in which the Gaussian Grid algorithm yields an accurate initial estimate for a least squares fitting method, resulting in a reduced number of iterations and hence reduced computational cost.

2.1 Introduction

Accurate attitude determination is a crucial aspect in many spacecraft missions. Several sensors to determine the attitude of a spacecraft have been developed, of which the star tracker is the most accurate. [133] This sensor estimates the attitude of the spacecraft by comparing star positions in an image taken on board of the spacecraft with a database of known star positions. The attitude determination error of a star tracker depends greatly on the accuracy of the algorithm that estimates the centroids of the stars in the focal plane, referred to as the centroiding algorithm [129]. To facilitate the determination of these centroids with subpixel accuracy, the optics of the star tracker are slightly defocused so that the star light is spread out over several pixels [75].
A variety of different algorithms to determine star centroids has been developed. An overview of the most common of these algorithms is given in [124] and [131]. The most accurate of these centroiding algorithms rely on fitting a point spread function (PSF) to the measured pixel data [7]. This point spread function is the impulse response of the imaging system [48] and can be very accurately modeled by a Gaussian profile [135]. While these algorithms determine the star centroid with great accuracy, they are computationally expensive because they use an iterative least squares function fitting approach [124]. Another approach is to calculate the center of gravity or variations on this center of gravity [131] to determine the star centroids. Algorithms using this approach are computationally considerably less expensive, but calculate the centroids with lower accuracy [124].
In this paper, a new centroiding algorithm, referred to as the Gaussian Grid algorithm, is developed. This algorithm uses explicit expressions to fit a Gaussian point spread function to the data. With respect to the existing algorithms that iteratively fit a Gaussian PSF to the data, the Gaussian Grid algorithm is significantly faster, but slightly less accurate. This decrease in accuracy is the result of approximations made in deriving the explicit expressions. Compared to the simple center of gravity methods, the PSF fitting of the Gaussian Grid algorithm greatly increases the accuracy, while the use of explicit expressions keeps the computational cost low. The Gaussian Grid algorithm is therefore a suitable method to be used in spacecraft that require accurate attitude estimation, but have little computational power.

Another interesting application of the Gaussian Grid algorithm is when it is used to provide accurate starting values for the methods using least squares fitting. While these algorithms yield high accuracy, one of their downsides is that they need an initial estimate of the parameters. A more accurate guess of these parameters can decrease the number of iterations needed to perform the fit and hence decrease the computational cost of the algorithm. In this

chapter, a hybrid algorithm is proposed in which the Gaussian Grid algorithm determines an accurate initial estimate of the parameters. These more accurate starting parameters result in a lower number of iterations and significantly reduce the computational cost as compared to the traditional least squares fitting algorithms. The hybrid algorithm yields the same accuracy as the existing least squares fitting method at a greatly reduced computational cost.

The Gaussian Grid algorithm allows to significantly reduce the computational cost of the attitude determination, while keeping the centroiding accuracy high. The increased efficiency allows to obtain the attitude information at a higher rate or use more stars in the star tracking procedure, this way increasing the performance of the attitude determination and control system. The novel algorithm can also reduce the computational cost of the spacecraft bus, allowing to carry a more demanding payload or lower the cost of the spacecraft.

First, the input for the algorithms, being the star images with their noise sources, is presented. Then, the state of the art algorithms which will be compared to the novel algorithm are discussed. After that, the novel Gaussian Grid algorithm is derived, followed by an analysis of the algorithm. The following section describes the hybrid method in which the Gaussian Grid algorithm calculates starting values for the least squares fitting approach. After that, the test setup is discussed. Finally, the results of the proposed algorithms and the state of the art algorithms are shown and discussed.

2.2 Stars and Noise

2.2.1 The Star Signal

A star at a large distance can be regarded as a point source. In order to facilitate the determination of the star centroid, the optics of the star tracker are slightly defocused to ensure that the light of the star is spread out over several pixels. For an ideal lens with a circular aperture, the resulting point spread function can be described by an Airy Disk. A good approximation of this Airy Disk is an elliptical Gaussian [135]. The resulting signal of a star on a pixel with coordinates (x,y) can then be computed as [129]:

$$V_{x,y} = N_e \int_{x-0.5}^{x+0.5} \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x-x_b)^2}{2\sigma_x^2}} \int_{y-0.5}^{y+0.5} \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{(y-y_b)^2}{2\sigma_y^2}}, \tag{2.1}$$

where $V_{x,y}$ is the measured pixel value at coordinates (x, y), N_e is the total number of electrons captured by the detector, the centroid of the star is at coordinates (x_b, y_b), and σ_x and σ_y are the x and y standard deviations.

2.2.2 Noise Sources

In this section, the most important noise sources in a star tracker image are discussed.

Dark Current

Even without the presence of a light source, electrons can still be freed from the valence band in the silicon and be transferred to the pixel well, because of the non-zero temperature of the detector [129]. This dark current increases with temperature and was modeled by adding a constant signal to the image: DC. In reality, the Dark Current Non-uniformity will lead to a non-constant signal which is often described by a Poission distribution [8]. This was not modeled in the simulation.

The presented centroiding algorithm could in fact be used to partly mitigate the effect of Dark Current Non-uniformity. The method can provide all parameters of the expected point spread function. The difference between the measured pixel values and the modeled Gaussian pixel values can be used to generate a pixel map that holds for each pixel the expected noise value. Pixels with high noise will e.g. always have a higher difference between the modeled and measured values.

Shot Noise

Shot noise, which is also referred to as photon noise, is a source of noise that originates from the discrete nature of photons coming from a star. These photons arrive in discrete packages according to a Poisson distribution [129]. This noise source is the dominant temporal noise for bright stars and has the following standard deviation:

$$\sigma_{sn,i,j} = \sqrt{V_{i,j}}. \tag{2.2}$$

Shot noise was added to the images as random normal distributed noise with a standard deviation equal to the square root of the signal generated by the dark current and starlight.

Read Noise

Read noise is a combination of noise from the charge transfer within the detector and noise from the read-out amplifiers [129]. It is a constant, uncorrelated temporal noise [56] and was modeled as noise with a normal distribution, with zero mean and a constant standard deviation: σ_{ro}.

Quantization

Since any analog-to-digital converter has a finite number of bits, the measured signal needs to be rounded. This was modeled as follows:

$$N_{levels} = 2^{N_{bits}} - 1, \tag{2.3}$$

where N_{bits} is the number of bits. The signal was then rounded using equation (2.4):

$$Vr_{ij} = \text{round}\left(V_{ij}\frac{N_{levels}}{FWC}\right)\frac{FWC}{N_{levels}}. \tag{2.4}$$

In this equation, Vr_{ij} is the rounded signal, and the function *round* is used to round its argument to the nearest integer.

2.3 State of the art Algorithms

In this section, the most common state of the art centroiding algorithms are discussed. The first algorithms are based on a calculation of the center of gravity while the last two rely on calculating a functional fit to the data.
Before the centroiding algorithms are used, the star tracker images are preprocessed to detect stars in the image and to coarsely determine their position. The position of the pixel holding the largest pixel value in that area is often taken as a coarse estimate of the position. Next, 'clusters' of star data

are selected around the star. These clusters are square fragments of the image that are selected around the coarse estimate of the star. In this paper, the algorithms will be tested with clusters of size 3×3 pixels, 5×5 pixels, 7×7 pixels and 9×9 pixels. These clusters of data are the input for the centroiding algorithms.

2.3.1 Center of Gravity

The Center of Gravity method, which is also referred to as the Moment method [124], is calculated as shown in equation (2.5):

$$(x_b, y_b) = \left(\frac{\sum_{ij} I_{ij} x_{ij}}{\sum_{ij} I_{ij}}, \frac{\sum_{ij} I_{ij} y_{ij}}{\sum_{ij} I_{ij}} \right). \tag{2.5}$$

This is the fastest method to calculate star centroids. An important disadvantage is its sensitivity to high background noise [7], which reduces the accuracy[124]. This negative effect is limited when the background noise can be filtered out to a great extent.

2.3.2 Weighted Center of Gravity

The Weighted Center of Gravity method [131] is related to the Center of Gravity method, but incorporates a weighting function W_{ij} that takes into account the point spread function of the star tracker optics. Using this method, the centroid is determined with the following formula:

$$(x_b, y_b) = \left(\frac{\sum_{ij} W_{ij} I_{ij} x_{ij}}{\sum_{ij} W_{ij} I_{ij}}, \frac{\sum_{ij} W_{ij} I_{ij} y_{ij}}{\sum_{ij} W_{ij} I_{ij}} \right), \tag{2.6}$$

where a Gaussian function is used as the weighting function, since the point spread function of a circular aperture can be approximated by a Gaussian:

$$W(x, y) = a \, exp\left[- \left(\frac{(x - x_b)^2}{2\sigma_x^2} + \frac{(y - y_b)^2}{2\sigma_y^2} \right) \right]. \tag{2.7}$$

In this weighting function, the magnitude a is set equal to the maximum intensity value, (x_b, y_b) are chosen to be the coordinates of the pixel with the maximum intensity value, and the standard deviations (σ_x, σ_y) are determined by first calculating the full width at half maximum; $fwhm$. This is the width of the Gaussian function between the points where the function value is half of the maximum function value. This value is approximated by taking the square root of the number of pixels which have an intensity value higher than half the maximum value. The standard deviations (σ_x, σ_y) are then considered to be equal to each other and are calculated by using equation (2.8):

$$\sigma = \frac{fwhm}{2\sqrt{2log\,(2)}}. \tag{2.8}$$

Alternatively, the $fwhm$ can be taken as a fixed value, or as a varying function based on the position in the field of view, calculated with ray tracing modeling of the optics.

The Weighted Center of Gravity method is slower than the Center of Gravity method because it requires more calculations and function evaluations to determine the weighting function. This method can generally determine the centroid with higher accuracy because it uses information about the point spread function. If the initial estimate of the centroid in the weighting function is far off however, the incorporation of the weighting function may not improve or will even decrease the accuracy.

2.3.3 Iterative Weighted Center of Gravity

The Weighted Center of Gravity method [131] does not yield better results when the initial estimate of the centroid (x_b, y_b) is far off. To solve this issue, the Iterative Weighted Center of Gravity method computes the centroid of the function iteratively and updates the weighting function in each step. The determination of the centroid is then calculated in the n^{th} step as:

$$(x_b^n, y_b^n) = \left(\frac{\sum_{ij} W_{ij}^n I_{ij} x_{ij}}{\sum_{ij} W_{ij}^n I_{ij}}, \frac{\sum_{ij} W_{ij}^n I_{ij} y_{ij}}{\sum_{ij} W_{ij}^n I_{ij}} \right), \tag{2.9}$$

with weighting function:

$$W^n(x,y) = a\,exp\left[-\left(\frac{\left(x-x_b^{n-1}\right)^2}{2\sigma_x^2}+\frac{\left(y-y_b^{n-1}\right)^2}{2\sigma_y^2}\right)\right]. \tag{2.10}$$

The initial weighting function uses the same parameters as in the Weighted Center of Gravity approach. The iterative process is stopped when the centroid location does not change more than a predetermined threshold or when a maximum number of iterations has been reached. This process is significantly slower than the Weighted Center of Gravity method, since it uses an iterative approach. The accuracy of this method is generally higher than that of the other Center of Gravity methods.

2.3.4 Least Squares Gaussian Fit 1D

In this method, the x-coordinate and y-coordinate of the centroid are determined separately by fitting a one-dimensional Gaussian to the marginals [7] of the data. For the x-marginal for example, the intensity data with the same x-coordinate is summed:

$$V_{mx_i} = \sum_{j=-n_b}^{n_b} V_{x_i,y_j}. \tag{2.11}$$

In this equation, V_{mx_i} is the sum of the pixel values with the same x-coordinate, and is referred to as the x-marginal, V_{x_0,y_0} is the center coordinate of the cluster and n_b is calculated as:

$$n_b = \frac{n_{pr}-1}{2}, \tag{2.12}$$

where n_{pr} is the number of pixels on a row or column of the pixel cluster.

The least squares problem is then defined as:

$$\text{min. } S(\beta) = \sum_{i=-n_b}^{n_b} \left[V_{mx_i} - f(x_i,\beta)\right]^2 \tag{2.13}$$

where $\beta = (a, x_b, \sigma_x)$ is the vector with the parameters to be determined, and $f(x_i, \beta)$ is the function that will be fit through the data. This function is defined as:

$$f(x_i, \beta) = ae^{\frac{-(x_i - x_b)^2}{2\sigma_x^2}} \tag{2.14}$$

For the y-marginal, the equation is similar.

When the size of the cluster increases, it may occur that in the outermost pixels of the cluster, the signal-to-noise ratio becomes very small. In this case, the outermost pixel values that are added to the marginals may lower the accuracy instead of increasing it, because adding them mainly adds noise to the useful marginal values. To counteract this effect, a variation on the Least Squares Gaussian Fit 1D method is proposed in which the outermost pixel values are not added to the marginals. In this variation on the algorithm, only a reduced number of pixel values with the same x-coordinate is added to the x-marginal in equation (2.11). In other words, the value of n_b is lowered in equation (2.11). For a 9×9 cluster for example, it is possible to only take into account the five center rows to calculate the x-marginals. This variation on the Least Squares Gaussian Fit 1D method was also simulated in the tests and is referred to as the Least Squares Gaussian Fit 1D R(educed) method.

In this paper, the Levenberg-Marquardt algorithm [72] was used to determine the least-squares solution of this problem. This algorithm is an iterative procedure and needs an initial guess for the parameter vector β. Similar to the Weighted Center of Gravity; a is set equal to the maximum intensity value, (x_b, y_b) are chosen to be the coordinates of the pixel with the maximum intensity value and the standard deviations (σ_x, σ_y) are set equal to each other and are calculated using equation (2.8). The Levenberg-Marquardt algorithm provides a local minimum, not a global minimum. In tests, this proved not to be an issue since the initial guess is fairly accurate and there are generally no multiple minima in the region of the initial guess.

2.3.5 Least Squares Gaussian Fit 2D

The Least Squares Gaussian Fit 2D method fits an asymmetric Gaussian function through all the data points of the cluster. Wide field optics will typically have distortions that produce asymmetric PSFs at the edge of the field. While the axes of the Gaussian can typically rotate around the optical axis, in the methods

of this chapter, they are assumed to align with the pixel coordinates. The problem to be minimized using least squares is then:

$$\min \; S(\beta) = \sum_{i=-n_b}^{n_b} \sum_{j=-n_b}^{n_b} \left[V_{x_i,y_j} - f(x_i, y_j, \beta) \right]^2 \tag{2.15}$$

where $\beta = (a, x_b, y_b, \sigma_x, \sigma_y)$ is the vector with the parameters to be determined, n_b is calculated as in equation (2.12), V_{x_i,y_j} is the pixel value of coordinates (x_i, y_j), and $f(x_i, y_j, \beta)$ is the function that will be fit through the data. This function is defined as:

$$f(x_i, y_j, \beta) = a e^{\frac{-(x_i - x_b)^2}{2\sigma_x^2}} e^{\frac{-(y_j - y_b)^2}{2\sigma_y^2}}. \tag{2.16}$$

This least squares problem is solved using the Levenberg-Marquardt algorithm. The initial guess for β is the same as in the previous method and the values σ_x and σ_y are chosen to be equal to each other. This method is very accurate because it optimally fits a 2D Gaussian point spread function to the data. However, because of the iterative approach and the large number of parameters in β, it is a computationally intensive method.

2.4 Gaussian Grid Algorithm

In this section, the novel centroiding algorithm, refered to as the Gaussian Grid algorithm, will be presented. This algorithm uses explicit expressions to fit a Gaussian point spread function to the data.

The Gaussian Grid algorithm fits an elliptical Gaussian profile to the pixel data and determines the centroid of this Gaussian function to estimate the centroid of the star. To determine the parameters of the Gaussian function, the function is first transformed to remove the exponential functions (equation (2.17) to equation (2.18)).

$$V_{x_i,y_j} = a e^{\frac{-(x_i - x_b)^2}{2\sigma_x^2}} e^{\frac{-(y_j - y_b)^2}{2\sigma_y^2}}, \tag{2.17}$$

$$ln\left(V_{x_i,y_j}\right) = ln\left(a\right) - \frac{\left(\sigma'_y\left(x_i - x_b\right)^2 + \sigma'_x\left(y_j - y_b\right)^2\right)}{\sigma'_x\sigma'_y} \quad \text{with } \sigma'_x = 2\sigma_x^2\ , \sigma'_y = 2\sigma_y^2 \tag{2.18}$$

where V_{x_i,y_j} is the measured pixel value, a is the amplitude of the Gaussian, (x_i, y_j) are the coordinates of the pixel, (x_b, y_b) are the coordinates of the centroid of the Gaussian, and σ_x and σ_y are the x and y standard deviations. Next, the data is split up into rows of pixel values to calculate x_b, and columns of data to calculate y_b. The algorithm will be explained further using rows of data to calculate x_b, the procedure to calculate y_b is similar.

For each row of pixel data, the total difference between the function values of the Gaussian profile and the measured values is minimized (equation (4.9)). For the row where the y-coordinate is y_0, this is expressed as:

$$\min\ \Gamma_{GG}\left(a, x_b, y_b, \sigma'_x, \sigma'_y\right) =$$

$$\sum_{i=1}^{n_{pr}} w_i \left[ln\left(V_{x_i,y_0}\right) - ln\left(a\right) + \frac{\sigma'_y\left(\left(x_i - x_b\right)^2 + \sigma'_x\left(y_0 - y_b\right)^2\right)}{\sigma'_x\sigma'_y} \right]^2, \tag{2.19}$$

where n_{pr} is the number of pixels of one row of the considered pixel cluster, and w_i are weights given to the distances. These weights will be discussed in more depth in a following section.

By imposing that the measurement locations lie on an equally spaced grid - since the pixel locations form such a grid - this optimization can be greatly simplified. This is done by describing the measurement coordinates (x_i, y_i) relative to a central coordinate (x_c, y_c). This is depicted in Table 2.1 for a 3×3 pixel cluster.

Table 2.1: Measurement coordinates grid.

$(x_c - 1, y_c + 1)$	$(x_c, y_c + 1)$	$(x_c + 1, y_c + 1)$
$(x_c - 1, y_c)$	(x_c, y_c)	$(x_c + 1, y_c)$
$(x_c - 1, y_c - 1)$	$(x_c, y_c - 1)$	$(x_c + 1, y_c - 1)$

Because the problem is simplified by transforming the Gaussian function (equation (2.17) to equation (2.18)) and by expressing the pixel coordinates relative to the center pixel, it is possible to solve the optimization analytically. The unknown variables a, x_b, y_b, σ_x, and σ_y which minimize this cost function can be found by calculating the derivative of cost function Γ_{GG} to each of the unknowns, setting the five obtained equations equal to zero and solving this system of five equations.

This yields explicit relationships that allow to very efficiently calculate the centroid of the star. Another advantage is the fact that no initial estimate of the amplitude a and the standard deviations σ_x and σ_y are needed, as opposed to approaches using least squares solutions. As an example of the explicit relationships yielded by this analytical approach, the expression to calculate the x-coordinate of the star centroid, x_b, when a row of 3 pixels is used, is given below. This is for the case where the row with y-coordinate, y_c, was used:

$$x_b = x_c + \frac{B_{-1,0} ln\left(V_{x_c-1,y_c}\right) + B_{0,0} ln\left(V_{x_c,y_c}\right) + B_{1,0} ln\left(V_{x_c+1,y_c}\right)}{A_{-1,0} ln\left(V_{x_c-1,y_c}\right) + A_{0,0} ln\left(V_{x_c,y_c}\right) + A_{1,0} ln\left(V_{x_c+1,y_c}\right)} \quad (2.20)$$

$$= x_c + \frac{BV_{y_c}}{AV_{y_c}}. \quad (2.21)$$

The values $A_{i,j}$ and $B_{i,j}$ are calculated in the same way for each row, but are a function of the weights w_i given to each distance in the cost function. The expressions to calculate the values $A_{i,j}$ and $B_{i,j}$ are given in the appendix. Equation (2.21) is a shorter representation of equation (2.20), where BV_{y_c} is the sum of the elements in the nominator of equation (2.20), and AV_{y_c} is the sum of the elements in the denominator. As a final step, the x-coordinate of the centroid, x_b, is determined, based on the information of all the rows. This is done by first taking the sum of the nominators of equation (2.20) for each row, and dividing that by the sum of the denominators of each row. The result is added to x_c, to obtain the estimated x-coordinate of the centroid:

$$x_b = x_c + \frac{BV_{y_c-1} + BV_{y_c} + BV_{y_c+1}}{AV_{y_c-1} + AV_{y_c} + AV_{y_c+1}}. \quad (2.22)$$

This algorithm yields high accuracy, obtained by fitting an elliptical Gaussian PSF to the data, while being computationally inexpensive because explicit relationships to calculate the star centroid were determined.

2.5 Gaussian Grid Analysis

In this section, the new Gaussian Grid algorithm is analysed further. The approach of this algorithm is similar to that of the Least Squares Gaussian Fit 1D method. The Gaussian Grid method also fits a Gaussian function to rows and columns of data. However, the closed form expressions of the Gaussian Grid algorithm make it significantly more computationally efficient than the Least Squares Gaussian Fit 1D method.

In order to obtain these closed form expressions, the problem was simplified by taking the natural log of the measurement and model. This transformation affects the performance of the Gaussian Grid algorithm. In the first part of this section, the effect of the transformation on the noise will be examined. After this, there is a discussion on the weights that are used in both the Least Squares methods and the Gaussian Grid method.

2.5.1 Noise Characteristics

To obtain the closed form expressions of the Gaussian Grid algorithm, the natural logarithm of the measurement and model values was taken. This transformation also affects the noise and may reduce the performance of the algorithm. This effect is studied in this section, based on reference [22].

Using the state of the art least squares method, the parameter x_b is estimated by fitting a Gaussian function through data with noise $\nu_{x,y}$. This is assumed to be Gaussian white noise with mean zero and variance equal to σ^2:

$$V_{x,y} = ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}} + \nu_{x,y}. \tag{2.23}$$

The covariance of the estimate error can then be calculated using equation 2.24:

$$P_1 = \left(H_1^T \, \sigma^{-2} H_1\right)^{-1}, \tag{2.24}$$

where H_1 holds values for every pixel used in the optimization:

$$H_1 = \begin{bmatrix} \dfrac{a\left(x_1 - x_b\right) e^{\frac{-(x_1-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y_1-y_b)^2}{2\sigma_y^2}}}{\sigma_x^2} & \dfrac{a\left(x_2 - x_b\right) e^{\frac{-(x_2-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y_2-y_b)^2}{2\sigma_y^2}}}{\sigma_x^2} & \ldots \end{bmatrix}^T. \tag{2.25}$$

For the Gaussian Grid, the natural log is taken on both sides of equation (2.23). A first-order series is used to calculate the logarithm:

$$ln\left(V_{x,y}\right) \approx ln\left(a\right) - \frac{\left(2\sigma_y^2\left(x - x_b\right)^2 + 2\sigma_x^2\left(y - y_b\right)^2\right)}{2\sigma_x^2 2\sigma_y^2} + \frac{2\nu_{x,y}}{ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}} + \nu_{x,y}}. \tag{2.26}$$

The measurement noise is now not longer Gaussian. Using the binomial series expansion, the value of this noise, represented as $\epsilon_{x,y}$, can be calculated as follows:

$$\epsilon_{x,y} \equiv \frac{2\nu_{x,y}}{ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}} + \nu_{x,y}}; \tag{2.27}$$

$$\approx \frac{\nu_{x,y}}{ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}}} - \frac{\nu_{x,y}^2}{2a^2 e^{\frac{-(x-x_b)^2}{\sigma_x^2}} e^{\frac{-(y-y_b)^2}{\sigma_y^2}}}. \tag{2.28}$$

In the next step, the variance of $\epsilon_{x,y}$ is determined as:

$$\varsigma_{x,y}^2 = E\left\{\epsilon_{x,y}^2\right\} - E\left\{\epsilon_{x,y}\right\}^2; \tag{2.29}$$

$$= \frac{\sigma^2}{\left(ae^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}}\right)^2} + \frac{\sigma^4}{2\left(a\, e^{\frac{-(x-x_b)^2}{2\sigma_x^2}} e^{\frac{-(y-y_b)^2}{2\sigma_y^2}}\right)^4}. \tag{2.30}$$

The covariance of the estimate error is then given by:

$$P_2 = \left(H_2^T\ diag\left[\varsigma_{x_1,y_1}^{-2}\ \ \varsigma_{x_2,y_2}^{-2}\ \ ...\right] H_2\right)^{-1} \tag{2.31}$$

The H-matrix, H_2, now holds the following values:

$$H_2 = \left[\frac{x_1 - x_b}{\sigma_x^2} \quad \frac{x_2 - x_b}{\sigma_x^2} \quad ...\right]^T. \tag{2.32}$$

The covariance matrices of both approaches are equal when the component of $\varsigma_{x,y}^2$ holding the term with σ^4 is negligible. This will be the case for values with a high signal-to-noise ratio. When the signal-to-noise ratio decreases, this error component will become more important and the performance of the Gaussian Grid method as compared to the least squares method will deteriorate. The signal-to-noise ratio will be smaller when the measurements become less reliable. It will also become smaller in the outermost pixels, especially when using large clusters, because the signal values are small in those pixels. In these cases, a deterioration of the accuracy of the Gaussian Grid method compared to the least squares methods can be expected.

2.5.2 Weights

In this section, the optimal weights w_i are determined for the cost function of equation (4.9). In a least squares function fitting approach, the distance squared between the measurement values and the model values is minimized. With M the measurement value, and e the error between the measurement value and the value of the model Gaussian, the distance squared d can be written as:

$$d_1 = (M - (M - e))^2 = e^2 \tag{2.33}$$

If the measurement value is multiplied with a factor k, the distance squared remains the same:

$$d_2 = (k\,M - (k\,M - e))^2 = e^2. \tag{2.34}$$

In order to ensure that the distance squared d remains the same in the presence of the same error e, regardless of the measurement value, weights w can all be chosen equal to one:

$$d_1 = (M - (M - e))^2 = w.\,(k\,M - (k\,M - e))^2 = w.d_2; \tag{2.35}$$

$$e^2 = w.e^2; \tag{2.36}$$

$$w = \left(\frac{e}{e}\right)^2 = 1 \tag{2.37}$$

In the Gaussian Grid optimization however (equation (4.9)), the distance squared between the natural logarithm of the measurement values and the natural logarithm of the model values is minimized:

$$d_1 = (ln(M) - ln\,(M - e))^2 \tag{2.38}$$

In this case, the distance squared does not remain the same when the measurement value is changed, but the error e is the same:

$$d_2 = (ln(k\,M) - ln\,(k\,M - e))^2\,. \tag{2.39}$$

In this second case, with higher measurement values (with $k > 1$), the difference is smaller: $d_2 < d_1$. This means that the importance of the errors on larger pixel values is attenuated while the errors at lower pixel values will be more important in the optimization. This results in a less accurate solution. Therefore, weights

are chosen that will compensate this effect. These weights are chosen so that regardless of the measurement value, the distance squared will be the same if the error e is the same. This is stated in equation (2.40):

$$d_1 = \left(ln\left(M\right) - ln\left(M - e\right)\right)^2 = w.\left(ln\left(k\,M\right) - ln\left(k\,M - e\right)\right)^2 = w.d_2. \quad (2.40)$$

This equation can be transformed to:

$$ln\left(\frac{M}{M-e}\right) = \sqrt{w}\; ln\left(\frac{k\,M}{k\,M-e}\right), \quad (2.41)$$

and by further simplifying to find w:

$$w = \left(\frac{ln\left(1-\frac{e}{M}\right)}{ln\left(1-\frac{e}{k\,M}\right)}\right)^2. \quad (2.42)$$

Since the errors are small compared to the measurement values, the values in the logarithmic functions are close to 1. These functions can therefore be approximated by their first order Taylor approximation. In the case of a constant error e, which does not depend on the pixel value M, the weights can be determined using equation (2.43):

$$w = \left(\frac{\frac{e}{M}}{\frac{e}{k\,M}}\right)^2 = k^2. \quad (2.43)$$

In this case, the weight is calculated as the squared value of k, which is the ratio between the pixel value and a reference pixel value. By setting the reference pixel value equal to 1, the weights are easily calculated as the square of the pixel value. For the weight of the center pixel, $w_{0,0}$, for example, this becomes:

$$w_{0,0} = V_{x_c,y_c}^2 \quad (2.44)$$

The weights determined above are valid under the assumption that the error e does not depend on the measurement value. However, from equation (2.2),

it is clear that the shot noise depends on the measurement value. The error e therefore contains a constant component, and a component which is proportional to the square root of the measurement signal:

$$e = C + \sqrt{M} \tag{2.45}$$

With this adapted error e, the weights for the least squares methods are adapted from equation (2.37):

$$w = \left(\frac{C + \sqrt{M}}{C + \sqrt{k\,M}} \right)^2 \tag{2.46}$$

When the constant component of the error is significantly higher than the error which is dependent on the measurement value, the weights again become equal to one. When the shot noise is significantly higher than the other noise component, the weights become equal to $\frac{1}{k}$. By again setting the reference pixel value equal to 1, the weight for the center pixel would then become:

$$w_{0,0} = \frac{1}{V_{x_c,y_c}} \tag{2.47}$$

For the Gaussian Grid method, the weights with this adapted error become:

$$w = \left(\frac{\frac{C+\sqrt{M}}{M}}{\frac{C+\sqrt{k\,M}}{k\,M}} \right)^2 \tag{2.48}$$

When the constant component of the error is significantly higher than the error which is dependent on the measurement value, the weights again become equal to k^2. When the shot noise is significantly higher than the other noise component, the weights become equal to k. By again setting the reference pixel value equal to 1, the weight for the center pixel would become:

$$w_{0,0} = V_{x_c,y_c} \tag{2.49}$$

In the case where there is significant shot noise, the weights will reduce the effect of an error on the larger pixel values, because there is higher shot noise on

these values. These adapted weights will be used in the simulations. Dependent on the added noise in the simulations, the weights will be adapted as depicted in table 2.2.

Table 2.2: The weights used in the least squares methods and the Gaussian Grid method, depending on the noise sources.

Noise characteristics	LSQ methods	Gaussian Grid
shot noise $\ll$ other	$w_{0,0} = 1$	$w_{0,0} = V^2_{x_c,y_c}$
shot noise $\gg$ other	$w_{0,0} = \frac{1}{V_{x_c,y_c}}$	$w_{0,0} = V_{x_c,y_c}$

2.6 Hybrid Methods

The closed form expressions of the Gaussian Grid method allow us to obtain the star centroids with a very low computational time. The downside of this method is that it yields slightly lower accuracy than the most accurate least squares methods, because of the simplifications made in determining the closed form expressions. When the highest accuracy is required, it is therefore necessary to use the Least Squares Gaussian Fit 2D method. The downside of this method is that it is computationally complex and it requires an initial estimate of the parameters of the Gaussian function.

When the initial estimate of the Least Squares Gaussian Fit 2D method is inaccurate, the method will require more iterations to reach its optimal estimate, resulting in a greater computational time. Conversely, a very accurate initial estimate can greatly speed up the procedure. In the section below, a 'hybrid' algorithm is proposed in which the estimate of the Gaussian Grid method is used as the initial estimate for the Least Squares Gaussian Fit 2D method. A second hybrid algorithm improves the initial estimate of the Least Squares Gaussian Fit 2D method by calculating the centroids using the Center of Gravity method. This second hybrid method allows us to improve part of the initial estimate at a very low computational cost.

2.6.1 Hybrid Gaussian Grid Method

In order to obtain an initial estimate for the Least Squares Gaussian Fit 2D method, the Gaussian Grid method also needs to estimate the values of σ_x, σ_y and a. These values are, like the centroids, calculated from equation (4.9). The approach to calculate the standard deviations is similar to that of calculating the centroids. The expressions for determining σ_x are given below for one row of data:

$$\sigma_x = \frac{D}{C_{-1,0} ln\left(V_{x_c-1,y_c}\right) + C_{0,0} ln\left(V_{x_c,y_c}\right) + C_{1,0} ln\left(V_{x_c+1,y_c}\right)}; \tag{2.50}$$

$$\sigma_x = \frac{DV_{y_c}}{CV_{y_c}}. \tag{2.51}$$

When the rows are combined for a 3×3 cluster, σ_x is calculated as:

$$\sigma_x = \frac{DV_{y_c-1} + DV_{y_c} + DV_{y_c+1}}{CV_{y_c-1} + CV_{y_c} + CV_{y_c+1}}. \tag{2.52}$$

The expression to calculate σ_y is similar. The expressions to calculate the weights D and $C_{i,j}$ are given in the appendix. The equation to calculate the value of a is rather long and is therefore not shown here.

Using these expressions, all five parameters of the Gaussian function are determined with considerable accuracy. These parameters are then supplied to the Least Squares Gaussian Fit 2D method to serve as the initial estimate.

2.6.2 Hybrid Center of Gravity Method

The initial estimate of the Gaussian function parameters can be improved using the Center of Gravity method. This way, the values x_b and y_b in the initial estimate will have increased accuracy. While the Center of Gravity method is faster than the Gaussian Grid method, it does not allow to calculate values for the standard deviations and the amplitude of the Gaussian function. The values of σ_x, σ_y and a will therefore still be determined as done in the normal Least Squares Gaussian Fit 2D method. The initial estimate will therefore be determined faster than in the Hybrid Gaussian Grid method, but it will be less accurate, which may result in more iterations of the least squares method.

2.7 Test setup

For each test, 10 000 images containing one star were generated, using the procedure mentioned in section "Stars and Noise". Tests were performed with varying levels of noise and different camera characteristics. The number of electrons in the entire star signal was equal to the Full Well Capacity mentioned for the tests.

The images were first processed to select a cluster around the star. A coarse estimate of the star location was determined by selecting the maximum intensity value in the image. Once this location was found, clusters of 3×3, 5×5, 7×7 and 9×9 pixels were selected around it.

From the values in these clusters, a background signal was subtracted. Because of Dark Current and Read Noise, the sensor measures a non-zero signal even for a black background. To estimate the background signal, an algorithm was developed which selected a pixel area located a certain distance away from the star in the image and calculated the average signal in a 5×5 pixel cluster. This average value represents the background signal and was subtracted from the pixel values.

These clusters were then passed to the different centroiding algorithms. For the least squares methods, the C++ library *lmfit* was used. This library uses the Levenberg-Marquardt algorithm. This algorithm is an iterative procedure which starts from a user-supplied initial estimate. During simulations, the iterative procedure was stopped when the difference squared between the current estimate and the previous estimate, calculated in fractions of a pixels, was smaller than $1e^{-6}$ for both the x- and y-coordinate of the centroid.

The accuracy of the algorithms was assessed by calculating the Euclidean distance between the estimated centroid and the true centroid. This distance was calculated using equation (2.53):

$$D = \sqrt{(x_e - x_t)^2 + (y_e - y_t)^2}. \tag{2.53}$$

In this equation, (x_e, y_e) is the estimated centroid and (x_t, y_t) is the true centroid.

The speed of the algorithms was determined by logging the calculation time of 1.000.000 algorithm executions in order to get a good averaging effect. The

time was logged using the C++ command QueryPerformanceCounter. The tests were performed with a Dell Latitude laptop with i7 processors and 8Gb Ram.

In the Gaussian Fit Least Squares 1D R method, the number of rows used in calculating the x-marginal (and columns used in calculating the y-marginal) is reduced, because the outermost pixel values have low signal-to-noise ratios which can decrease the accuracy. In this reduced variation of the Gaussian Fit Least Squares 1D method, only the five center rows or columns were used to calculate the marginals when using a 7×7 and 9×9 pixel cluster. In other words, n_b was set equal to 2 in equation (2.11).

For the Hybrid Gaussian Grid algorithms, the values for σ_x and σ_y were calculated using a 5×5 cluster, also when a 7×7 and a 9×9 pixel cluster was used. To value of a was estimated using a 3×3 cluster for all simulations of this method. This was done because the expressions to calculate σ_x, σ_y and especially a become lengthy when the pixel cluster increases and therefore the computational time rises significantly.

2.8 Results

In this section, the accuracy and speed of the proposed centroiding algorithm is compared to that of the state of the art algorithms. First, the speed results are presented. After this, the accuracy results for three different scenarios with varying levels of noise are shown. On top of considering artificial scenarios, the results using the parameters [129] of the star trackers that will be used in ESA's Plato mission are shown.

2.8.1 Speed

The computational time of the different algorithms is given in μs in Table 2.3 and is visualised in Fig. 2.1.

Because of its simplicity, the fastest method is the Center of Gravity algorithm. The Weighted Center of Gravity method requires a lot of function evaluations which makes it computationally more complex. The computational time of the Iterative Weighted Center of Gravity method is of course an order of magnitude higher since it iteratively performs the calculations done in the Weighted Center

Table 2.3: The computational time to determine the centroid of one star (μs).

Method	3×3	5×5	7×7	9×9
Gaussian Grid	0.196	1.431	4.897	12.646
Center of Gravity	0.055	0.101	0.158	0.253
Weighted Center of Gravity	4.460	11.788	23.112	37.823
Iterative Weighted Center of Gravity	27.717	94.710	185.391	305.232
Gaussian Fit Least Squares 1D	39.347	55.546	77.285	99.089
Gaussian Fit Least Squares 1D R	39.360	55.469	70.896	83.780
Gaussian Fit Least Squares 2D	52.548	121.933	221.772	353.606
Hybrid Gaussian Grid	35.001	81.323	148.218	233.203
Hybrid Center of Gravity	48.023	102.762	185.601	299.413

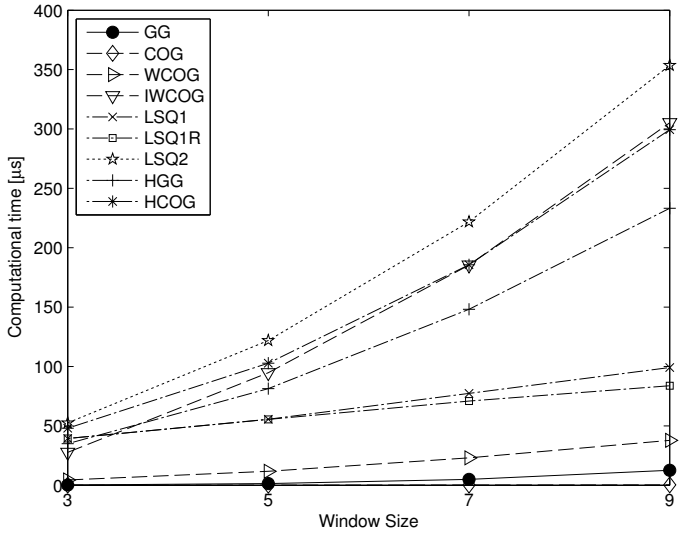


Figure 2.1: The computational time for the different algorithms with different cluster sizes.

of Gravity method. The least squares methods require a large computational time because they also estimate the centroid using an iterative procedure. The 1D Least squares method is faster because the parameter β in this case has less parameters to optimize. The Gaussian Fit Least Squares 1D R is slightly faster because the reduced number of data points lowers the computational cost.

The Gaussian Grid algorithm, which is presented in this paper, is the second fastest algorithm. This algorithm estimates the location of the star centroid using simple explicit equations, which makes it very efficient. When a 5×5 pixel cluster is used, the algorithm is almost 40 times faster than the Gaussian Fit Least Squares 1D method and more than 80 times faster than the Gaussian Fit Least Squares 2D method, which are the most accurate methods.

The effect of using an improved initial estimate in the hybrid methods is also clear. For the Hybrid Gaussian Grid method, the calculation time is reduced by 33 percent for a 5×5 pixel cluster, compared to the normal Gaussian Fit Least Squares 2D method. The Hybrid Center of Gravity method is also faster than the Gaussian Fit Least Squares 2D method, but the reduction in computational time is lower than for the Hybrid Gaussian Grid method. This is because the initial estimates of the Gaussian function parameters are estimated with a lower accuracy, resulting in a larger number of iterations in the least squares method.

From Table 2.3, it is clear that the Gaussian Grid method is a computationally efficient method. It is significantly faster than the most accurate least squares methods and is only slower than the simple Center of Gravity method in these simulations. Using the estimate of the Gaussian Grid as a starting estimate for the Gaussian Fit Least Squares 2D method clearly speeds up the procedure. This hybrid algorithm allows to obtain the high accuracy of the least squares approach at a substantially reduced computational cost.

2.8.2 Accuracy

In this section, the accuracy of the different centroiding algorithms is compared for three different scenarios with various noise levels. The first two scenarios are simulated scenarios with moderate and high noise values. The third scenario uses the values of the Plato mission star trackers.

scenario 1

The first scenario has **moderate noise levels**. The parameters used to calculate the noise sources are given below:

In this case, the level of Shot Noise is considerably smaller than the level of Dark Current and Read noise. Because of this, the weights are calculated as in the first row of table 2.2. typical sensors work with 10 or 12 bits, so an 8 bit system is a worst case assumption.

Table 2.4: Parameters used in the image generation of scenario 1.

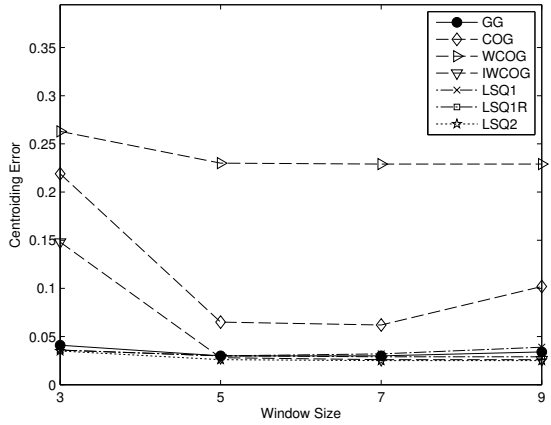
Parameter	Value
Full well capacity (FWC)	100 e3
DC	FWC/50
σ_{ro}	FWC/50
N_{bits}	8
(σ_x, σ_y)	(1.1, 1)

Table 2.5: The rms centroiding error of the different centroiding algorithms expressed in fraction of a pixel for scenario 1.

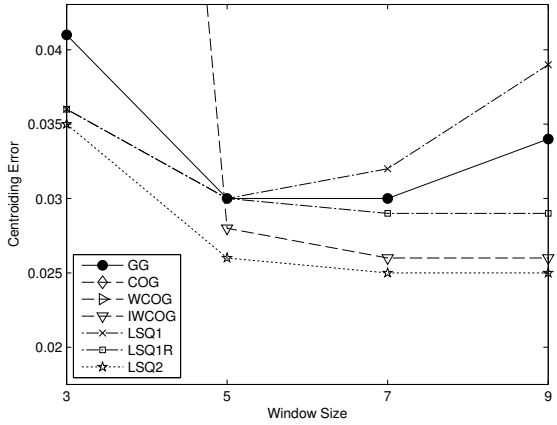
Method	3×3	5×5	7×7	9×9
Gaussian Grid	0.041	0.030	0.030	0.034
Center of Gravity	0.219	0.065	0.062	0.102
Weighted Center of Gravity	0.263	0.230	0.229	0.229
Iterative Weighted Center of Gravity	0.148	0.028	0.026	0.026
Gaussian Fit Least Squares 1D	0.036	0.030	0.032	0.039
Gaussian Fit Least Squares 1D R	0.036	0.030	0.029	0.029
Gaussian Fit Least Squares 2D	0.035	0.026	0.025	0.025
Hybrid Gaussian Grid	0.035	0.026	0.025	0.025
Hybrid Center of Gravity	0.035	0.026	0.025	0.025

The accuracy of the different algorithms is shown in table 2.5 and in Fig. 2.2, where the root mean square error is expressed in fraction of pixel size for the different methods and using different cluster sizes. Since the hybrid methods have the same accuracy as the Gaussian Fit Least Squares 2D method, only the accuracy of the latter is shown in the figure.

From these results, it follows that the accuracy of the Gaussian Grid method is significantly better than that of the Center of Gravity method. When the best results for both methods are compared, the rms error of the Center of Gravity method is more than 100% higher than the rms error of the Gaussian Grid method. The Gaussian Grid method obtains its best results already in the 5×5 pixel cluster, while the Center of Gravity method obtains it in the 7×7 cluster. The Center of Gravity method therefore requires more pixels to be read out from the sensor, which could well increase the computational cost of the entire



(a) Overall



(b) Detail of the accuracy results

Figure 2.2: The accuracy results showing the mean squared error in fraction of a pixel for scenario 1.

procedure to be higher than that of the Gaussian Grid method. The Gaussian Grid algorithm is less accurate for the 9 × 9 cluster than for the 7× 7 and 5 × 5 cluster. As was discussed in the section that analyses the Gaussian Grid algorithm, because of the transformation that is used in deriving the closed form expressions, an additional component is introduced in the effective noise, which reduces the accuracy when pixel values with a low signal-to-noise ratio are included. This is the case for the outermost pixel values of the 9 × 9 cluster.

The Weighted Center of Gravity method is less accurate than the Center of Gravity method because the weights decrease the accuracy when the estimate of the parameters of the weighting function is far off. The Iterative Weighted Center of Gravity method outperforms the Gaussian Grid method when the cluster is larger than 3×3. As discussed in the section presenting the state of the art algorithms, the accuracy of the Gaussian Fit Least Squares 1D method can decrease when the cluster size increases. Reducing the number of rows and columns used in calculating the marginals counteracts this decrease in accuracy, as can clearly be seen from the results of the Gaussian Fit Least Squares 1D R method. The best result is obtained by the Gaussian Fit Least Squares 2D method. The error of this method, and therefore also of the hybrid methods, is around 17 % lower than that of the Gaussian Grid method.

scenario 2

The second scenario has **high noise levels**. This scenario is representative for lower cost missions, where the star trackers are low-cost and are more prone to noise. The parameters used to calculate the noise sources are given below:

Table 2.6: Parameters used in the image generation of scenario 2.

Parameter	Value
Full well capacity (FWC)	100 e3
DC	FWC/25
σ_{ro}	FWC/30
N_{bits}	8
(σ_x, σ_y)	(1, 1.3)

In this case, the level of Shot Noise is considerably smaller than the level of Dark Current and Read noise. Because of this, the weights are calculated as in the first row of Table 2.2.

In the scenario with high noise values, the Gaussian Grid method again clearly outperforms the Center of Gravity method, as can be seen from table 2.7 and Fig. 2.3. The lowest obtained error of the Center of Gravity method is almost 80 % higher than that of the Gaussian Grid method. The Gaussian grid method yields its best result in a smaller cluster, which allows to read out less pixels from the sensor. The Weighted Center of Gravity method again produces inferior results while the Iterative Weighted Center of Gravity method has an rms error which is 12% lower than that of the Gaussian Grid method. The

Table 2.7: The rms centroiding error of the different centroiding algorithms expressed in fraction of a pixel for scenario 2.

Method	3×3	5×5	7×7	9×9
Gaussian Grid	0.085	0.052	0.052	0.063
Center of Gravity	0.255	0.103	0.089	0.143
Weighted Center of Gravity	0.288	0.239	0.236	0.236
Iterative Weighted Center of Gravity	0.196	0.051	0.044	0.044
Gaussian Fit Least Squares 1D	0.073	0.050	0.056	0.063
Gaussian Fit Least Squares 1D R	0.073	0.050	0.048	0.048
Gaussian Fit Least Squares 2D	0.068	0.044	0.042	0.042
Hybrid Gaussian Grid	0.068	0.044	0.042	0.042
Hybrid Center of Gravity	0.068	0.044	0.042	0.042

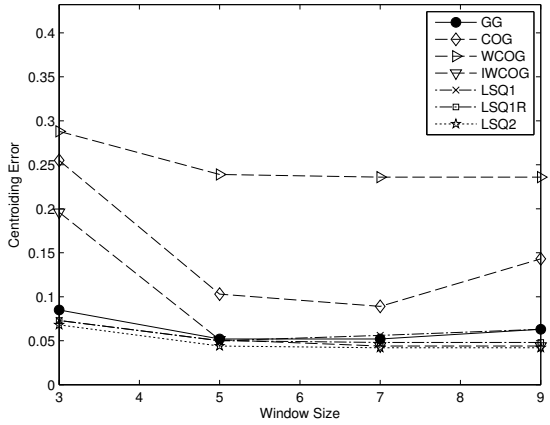
Gaussian Fit Least Squares 1D method yields better results for cluster sizes up to 5 × 5 pixels, for higher cluster sizes the accuracy again deteriorates. The Gaussian Fit Least Squares 1D R method does not deteriorate and instead yields slightly higher accuracy for the 7 × 7 and 9 × 9 cluster sizes. The most accurate method is again the 2D Least Squares method, of which the lowest rms error is 16 % lower than that of the Gaussian Grid method. The Hybrid methods have the same accuracy as the Gaussian Fit Least Squares 2D method.

scenario 3

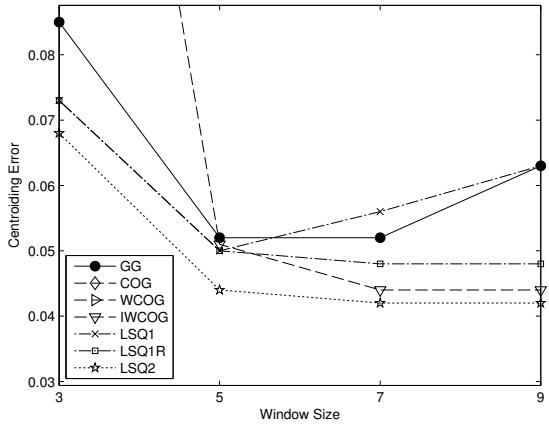
The third scenario uses the values of the fine guidance sensors of the **Plato mission** [129]. These star trackers have very low noise levels and produce very accurate attitude estimations:

Table 2.8: Parameters used in the image generation of scenario 3.

Parameter	Value
Full well capacity (FWC)	900 e3
DC	90
σ_{ro}	90
N_{bits}	16
(σ_x, σ_y)	(0.85, 0.85)



(a) Overall



(b) Detail of the accuracy results

Figure 2.3: The accuracy results showing the mean squared error in fraction of a pixel for scenario 2.

In this case, the Shot Noise is dominant over the Dark Current and Read noise. Because of this, the weights are calculated as in the second row of Table 2.2.

In this scenario with a high accuracy sensor, the Weighted Center of Gravity method again performs poorly because of the inaccurate estimate of the weights used in this method. The Iterative Weighted Center of Gravity performs better thanks to the iterations, but because the values of σ_x, σ_y and a used in the

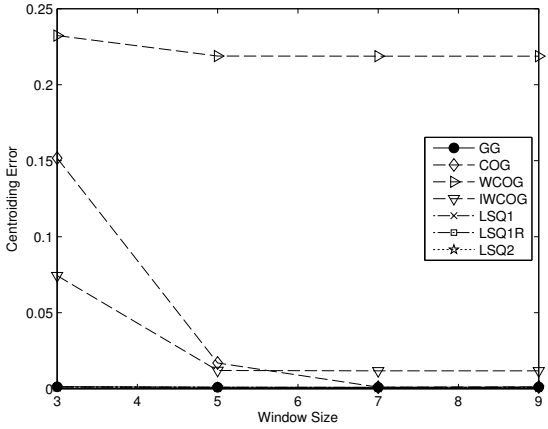
Table 2.9: The rms centroiding error of the different centroiding algorithms expressed in fraction of a pixel for scenario 3.

Method	3×3	5×5	7×7	9×9
Gaussian Grid	0.00127	0.00094	0.00089	0.00110
Center of Gravity	0.15166	0.01696	0.00100	0.00118
Weighted Center of Gravity	0.23234	0.21886	0.21880	0.21880
Iterative Weighted Center of Gravity	0.07438	0.01197	0.01174	0.01174
Gaussian Fit Least Squares 1D	0.00118	0.00088	0.00088	0.00091
Gaussian Fit Least Squares 1D R	0.00118	0.00088	0.00082	0.00082
Gaussian Fit Least Squares 2D	0.00118	0.00088	0.00082	0.00082
Hybrid Gaussian Grid	0.00118	0.00088	0.00082	0.00082
Hybrid Center of Gravity	0.00118	0.00088	0.00082	0.00082

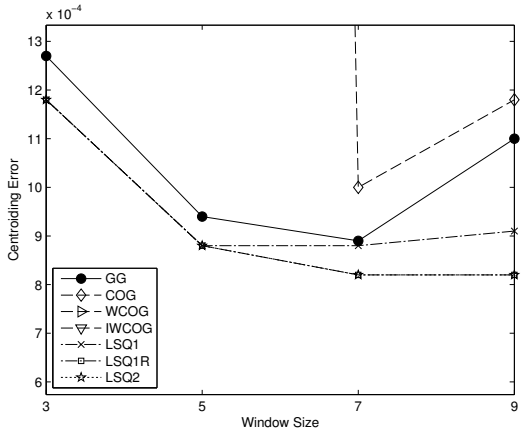
weights are still coarse estimates, the accuracy is limited. The Center of Gravity method yields low accuracy for small cluster sizes, but for the 7 × 7 cluster, the accuracy is quite high, although the mean square error is still around 22 % higher than the highest accuracy obtained by the Gaussian Fit Least Squares 2D method. The 1D and 2D Gaussian Fit Least Squares methods have similar accuracy in this scenario. The normal 1D method again loses accuracy when the cluster size increases, but the Gaussian Fit Least Squares 1D Reduced method yields the same accuracy as the Gaussian Fit Least Squares 2D for every pixel cluster. The Gaussian Grid method is slightly less accurate than the least squares methods but again more accurate than the Center of Gravity method. The lowest obtained mean square error is around 8% higher than that of the Gaussian Fit Least Squares 2D method.

2.9 Conclusion

In this paper, a novel star centroiding algorithm, referred to as the Gaussian Grid algorithm, was proposed and compared to state of the art centroiding algorithms. The Gaussian Grid algorithm uses closed form expressions to fit a Gaussian point spread function to the pixel data. This way, high accuracy is yielded while the computational cost remains low. In tests it was shown that the accuracy of the Gaussian Grid algorithm was significantly higher than that of the fastest state of the art algorithms that calculate a center of



(a) Overall



(b) Detail of the accuracy results

Figure 2.4: The accuracy results showing the mean squared error in fraction of a pixel for scenario 3.

gravity to determine the star centroid, while the computational cost remains low. This makes the Gaussian Grid algorithm a valuable option for spacecraft with limited computational power that require accurate attitude determination. Furthermore, a "hybrid" algorithm was presented in which the Gaussian Grid algorithm calculates the initial estimate for the most accurate Gaussian Fit Least Squares method. The more accurate initial estimate results in a reduced number of iterations needed in the least squares method. It was shown in

tests that this greatly reduces the computational cost of the entire approach. The resulting hybrid algorithm yields the same accuracy as the most accurate Gaussian Fit Least Squares method, at a lower computational cost. This reduction in computational cost can allow to use more stars in the star tracker algorithms, or acquire the attitude estimates at a higher rate, leading to an improved performance of the attitude estimation.

As mentioned, wide field optics will typically have distortions that produce asymmetric PSFs at the edge of the field. The axes of the modeled Gaussian will typically not align with the coordinate axes, as was assumed in this chapter and in a major part of the literature. A possibility could be to again use a hybrid approach, in which the result of the Gaussian Grid algorithm is used as input to a Least Squares method that does take into account the rotation in its cost function.

Chapter 3

Highly Robust Lost-in-space Algorithm Based On The Shortest Distance Transform

The contents of this chapter were published in the Journal of Guidance, Control and Dynamics: Delabie, T., Durt, T., Vandersteen, J. (2013). Highly Robust Lost-in-space Algorithm Based On The Shortest Distance Transform. *Journal of Guidance, Control and Dynamics*, 36 (2), 476-484.

A robust and fast algorithm that solves the lost-in-space problem for star trackers is presented in this chapter. The algorithm is based on an image processing technique, the Shortest Distance Transform, which transforms the camera image into a 2D look-up table. The information from the database can then be efficiently inserted into this table to compare the camera image with the database. This approach results in an algorithm which is robust to false stars, distortions on star positions and failure of registration of bright stars. As an example, the algorithm determines over 99% of camera images correctly when 400 false stars are added, distortions of 300 (1σ) arc seconds are present and the brightest star is missing in the image. In case of incorrect determination, a very reliable criterion indicates that the determination step has to be repeated.

3.1 Introduction

In order to point a payload to a target or an antenna to a ground station or to orient the solar panels toward the sun, a satellite needs to know its orientation in space. Several sensors have been developed to determine this orientation, of which the most accurate sensor is the star tracker. By taking a picture of the surrounding star panorama and comparing it to a database of known star positions, this sensor can typically determine the attitude of the satellite with an accuracy in the range of a few arc seconds [134].
Second generation star trackers can solve the lost-in-space problem, which means they can determine the attitude of the satellite without prior knowledge. After having done this, the star tracker switches to its tracking mode in which it can limit the database search by using prior knowledge of the attitude. The lost-in-space algorithm is used to initially find the attitude or to restore the attitude after it has been lost. A loss of attitude can occur after a power breakdown or because a fast maneuver, radiation or a bright object like the sun impeded the correct functioning of the tracking mode.
Several algorithms to solve the lost-in-space problem have been proposed. A good survey of the most common algorithms is given by Spratling and Mortari [120]. The great majority [107, 73, 6, 108, 55] of the existing algorithms uses features extracted from triplets of stars in the camera image and matches these to a preprocessed database of startriplet features. The most commonly used features are the inter-star angle, the magnitude of the stars and the area of the formed triangle [120]. Recently, several attempts to solve the lost-in-space problem with a neural networking approach have been made [59, 3], but the high complexity and the massive parallel architecture that is required to solve the algorithm currently limits the practical implementation of this approach.
A common shortcoming of the current algorithms is their weak robustness to distortions in the acquired camera image. Variations in magnitude because of badly functioning pixels, distortions on acquired star positions in the camera image caused by flaws in the optics or 'false stars' in the image evoked by radiation, cause a change in camera image features which results in an incorrect attitude estimation. Because of this weak robustness, current star trackers need high quality, expensive optics and detectors and are not well suited to operate in hostile radiation environments.
A highly robust star tracker will permit satellite manufacturers to acquire high accuracy attitude determination even in hostile environments. Furthermore, because of the high robustness of the algorithm, costs can be cut back on a hardware level. A considerable part of the budget normally needs to be invested into making the star tracker components radiation hardened, to limit the number of false stars which would otherwise compromise correct star identification.

Because of the very high robustness of this algorithm to false stars, this cost can be eliminated when the budget is limited, e.g. in small satellite projects. The presented, highly robust algorithm will allow this growing small satellite market to obtain accurate attitude information at low cost, thereby allowing this platform to handle a wider variety of missions.

In this chapter, a new lost-in-space algorithm with very high robustness is proposed. In section 3.2, an overview of the algorithm is given. The Shortest Distance algorithm was extensively tested with a variety of distortions in the camera image. The results of these tests are presented in section 3.3. This section also discusses the validation criterion. Finally, section 3.4 gives a conclusion of the presented work.

3.2 The Algorithm

This section starts with a discussion of the distance transform technique which is used to process the camera image. Next, the method to generate database images from the star catalogue is explained. After this, an explanation of how the images are matched onto each other is given, followed by a discussion of the image comparison. The methods to further improve the speed of the algorithm are then presented. At the end of this section, a schematic overview of the algorithm is given.

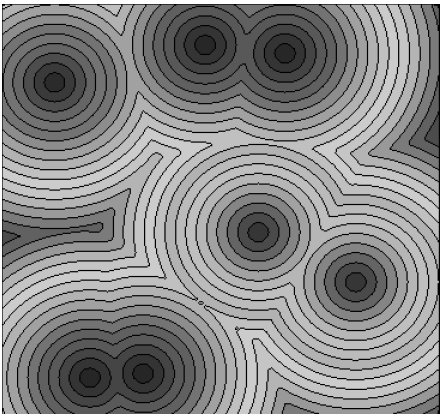
3.2.1 Distance Transform

A distance transformation - D - is a mathematical transformation used in image processing. The main idea of a distance transformation is simple: for each point of the entire set (Ω), the distance is computed to a certain subset (Ω^c) of it. This definition is given by equation (3.1).

$$D(p) = min\{d(p,q) \mid q \in \Omega^c\} \tag{3.1}$$

In this equation, p is the point of which we want to calculate the Shortest Distance value, q represents the points that are inside the subset (Ω^c). In the presented algorithm, the points inside the subset are the pixels which hold the star centroids. These centroids of the stars in the camera image were estimated in the star detection algorithm, which precedes the lost-in-space algorithm. By

(a) Simulated Star Tracker image



(b) Contour plot of the image

Figure 3.1: A picture of the stars (a) and a contour plot of the distance transformation map (b).

performing a distance transformation (DT), a map is created that contains for each pixel the distance to the nearest pixel which holds a star centroid. The method we use to calculate the distance is the Euclidean distance squared given in equation (3.2).

$$d(p,q) = (p_x - q_x)^2 + (p_y - q_y)^2, \tag{3.2}$$

The main advantage of this distance is that it is a symmetrical distance and therefore it can be used to compare images both translated and rotated. Because the squared Euclidean distance is taken, the distance transformation map consists of integers. This makes the computations faster without losing information.

A picture of a star panorama and the contour plot of its corresponding distance transformation is presented in figure 3.1.

Several algorithms to calculate the distance transformation have been developed. A comparative survey of the most frequently used methods is given by Fabbri et al. [38]. In our algorithm we used the algorithm developed by Maurer et al. [83]. This is an efficient linear-time algorithm that returns a map containing in each point the Euclidean distance squared to the closest star. Information from the processed database will be inserted into this map to compare the images. The processing of the database will be discussed next.

3.2.2 Database images

The data of the camera image needs to be compared to the data in a star catalogue, in this case the Hipparcos catalogue [36]. To be able to do this, images with the same field of view as the camera image are generated from the database. To effectively search the entire database, the generated images are evenly spaced over the database. Since our database represents a sphere of stars in an earth-centered coordinate system, this even spacing can be achieved by distributing points evenly over a sphere and using these distributed points as centers of the database images. After the centers have been selected, the images are generated by selecting all the stars which lie within the field of view (FOV) of the database image.

This database is reduced so that it only holds the magnitude and coordinates of the stars with a visual magnitude below 5.3. This magnitude was determined during simulations to make sure there are always three stars within the field of view of $20° \times 20°$ of the camera. Because the stars are not uniformly distributed over the sky (figure 3.2), one can not simply calculate such a threshold magnitude. In order to determine the threshold magnitude, a simulation was done using images created at 75,000 points distributed evenly over the sky. Since each image minimally needs to contain three stars to allow for a correct attitude determination, the magnitude of the third brightest star was withheld. Over the entire sky, this magnitude was 5.21. A margin was taken into account for distortions which leads to the chosen magnitude of 5.3.

Distributing points evenly on a sphere

Several methods to distribute points evenly on a sphere have been developed. The *packing method* tackles the problem of distributing points on a sphere so that the minimal distance between the points is maximized. The *covering method* minimizes the maximal distance of any point on the sphere to the closest one of the other points. The *minimal energy method* uses the model of electrons repelling each other and minimizes the Coulomb potential [93]. In this algorithm we use the covering method (figure 3.3) to ensure that the worst case difference between the camera image center and the database image center is minimal [57].

Constructing the database image

The generated points serve as centers for the database images. All the stars which lie within the field of view of the camera (γ) and which are brighter

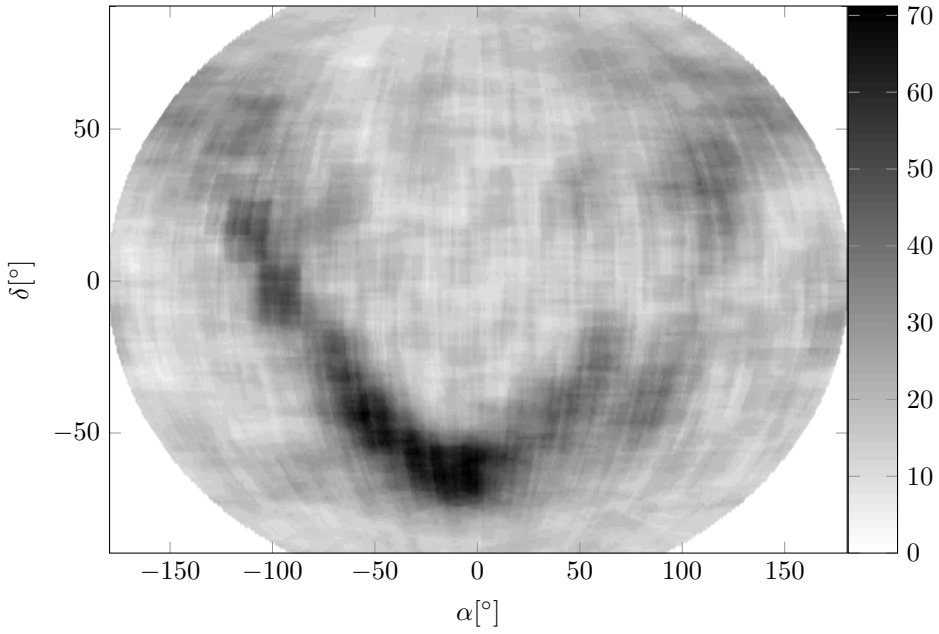


Figure 3.2: Distribution of clear stars (up to magnitude 5.3) in the night sky, Wagner VI projection. Notice the Milky Way.

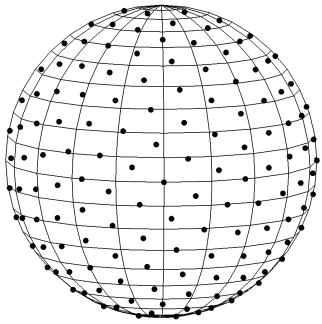


Figure 3.3: An example of an even distribution of points on a sphere.

than the threshold magnitude are subsequently selected out of the database. The generated database thus depends on the field of view of the camera and the threshold magnitude. Taking each star that has its right ascension and declination within $\frac{\gamma}{2}$ degrees of the center would not produce a good solution.

As can be seen by examining the grid of figure 3.3, such a partition would result in images that are a lot smaller at higher declination. A problem also arises when an image wraps around the poles. To overcome these two problems, the database is rotated around the negative right ascension and declination of the center of the image. This results in a database where the center of the image has right ascension and declination of 0 degrees. This eliminates the problems that arise when wrapping around the poles. Since the right ascension lines lie closer together at higher declination, the right ascension boundary has to be higher at a higher declination. It can be found that the boundary can be determined by equation (3.3)

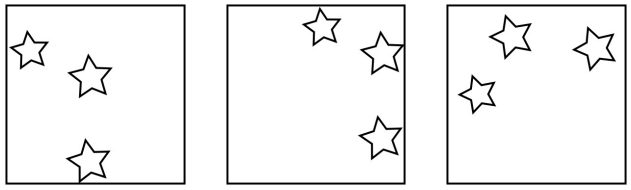
$$\alpha_b = \frac{\gamma}{2cos(\delta)} \tag{3.3}$$

The right ascension boundary is a function of the declination.

3.2.3 Matching of the camera image with the database images

At each of the evenly distributed points (section 3.2.2), a database image is generated (as in section 3.2.2). If there is an estimate of the attitude, the database images can be selected in the matching step, based on their distance to the estimated attitude. This way, the database images generated closest to the estimated attitude can be selected first. Since there is no prior knowledge or estimate of the attitude in the lost-in-space case, the database images are not ordered and are selected in order of increasing right ascension. In general, the center of the database image will not coincide with the center of the camera image (figure 3.4b). In most cases, the camera image will also be rotated in reference to the database images (figure 3.4c).

This is a problem since the camera image and database image need to have their star centroids on the same position to offer a positive determination. We call the act of rotating and translating the database image so that the star centroids of two equal images coincide, 'matching' the images. One method discussed in [19], using the two brightest stars and one new method we developed ourselves will be presented and compared to each other.

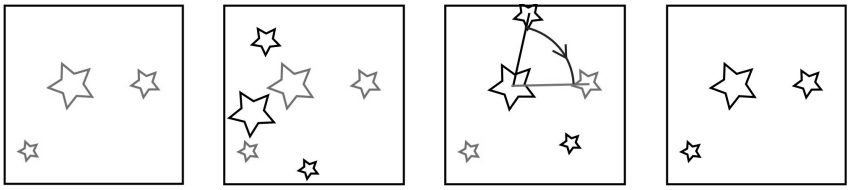


(a) camera image (b) different center (c) different rotation

Figure 3.4: Equal images with different orientation or position.

Two brightest stars method

In this method, the brightest star in the database image is placed at the same position as the brightest star of the camera image, thus solving the translation problem. To solve the problem of the different rotation, the second brightest star is rotated towards the second brightest star of the camera image. This process is illustrated in figure 3.5.
In figure 3.5a we see the image taken by the camera. The image that is extracted from the database is added to this in figure 3.5b. After the translation, the two brightest stars are on the same place as can be seen in figure 3.5c. Finally, the rotation step matches the images in figure 3.5d.



(a) (b) (c) (d)

Figure 3.5: The two brightest stars method.

Centroid method

This method uses the centroid of a certain number of brightest stars to solve the translation problem. Vectors between the centroid and the used brightest stars are calculated and the angle between them is then calculated. The smallest angle is used to solve the rotation problem. This process is depicted in figure 3.6.

The camera image is presented in figure 3.6a, and the database image is added to this in figure 3.6b. The centroids of the three brightest stars are represented by the small circle and are coincided in figure 3.6c. The vectors between the centroid and each of the three brightest stars are calculated and the three angles between those vectors are calculated in figure 3.6d. The smallest angle is selected. This smallest angle is used to rotate the stars as can be seen in figure 3.6e. The procedure results in a matching of center and roll angle of the images, as can be seen in figure 3.6f.

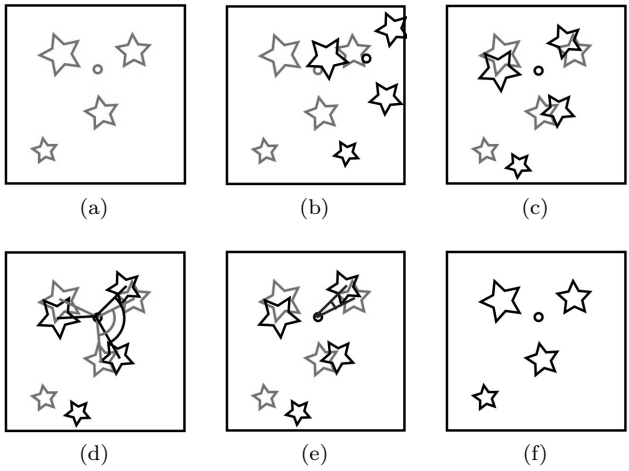


Figure 3.6: The centroid method.

Comparison of the methods

An important issue with matching the images is the fact that the order of brightness of the stars in the camera image and the corresponding database image might differ. This can happen when there are bright false stars or when the star detection algorithm outputs inaccurate values for the magnitude of the detected stars. Since a star tracker can determine star magnitude accurately down to an error of around 0.1 magnitude rms [1] and since the variation in magnitude between the brightest stars is generally a lot higher than this error, the order of brightness of the brightest camera image stars and database image stars is generally the same. However, to ensure correct matching even in the event that the order of brightness is different, the matching procedure for two images is performed a number of times while interchanging the order of brightness of a certain amount of brightest stars. The problem is that this results in a lot of combinations and hence a lot of iterations, which can greatly

slow down the algorithm. Therefore, the number of iterations should be reduced as much as possible without losing robustness.
The advantage of the second method is that the stars used to calculate the centroid can interchange their place in the brightness rank, without changing the location of the centroid. When for example three stars are selected out of the five brightest stars, this is translated to taking the combinations of 3 out of 5 (equation 3.5). For the first method, the relative brightness of the two stars in the set is important because the first star is used to tackle the translation problem and the second star to solve the rotation problem. This translates to taking the permutations of 2 out of 5. Because only the two first stars are important, this number is reduced (equation 3.4).

$$n_{two} = \frac{n!}{(n-2)!} \tag{3.4}$$

$$n_{centr} = \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3.5}$$

In table (3.1), the number of combinations and permutations is given. The first row gives the number of brightest stars which can be interchanged - n - the second row gives the number of permutations required by the two-star method. The last row gives the number of combinations required by the centroid method. In the centroid method, k is taken to be three.

Table 3.1: Comparison of number of iterations.

n	3	4	5	6	7
n_{two}	6	12	20	30	42
n_{cent}	1	4	10	20	35

Although matching the center and roll angle with the centroid method requires more calculations, it is clear that less combinations between stars are needed. Since the time to match the images is negligible compared to the time required for the comparison step, the computational time of the algorithm with the centroid method is one third lower when 6 brightest stars are used. This method is more robust than the two star method and faster, so this is preferred.

3.2.4 Image selection using the shortest distance transform map

Once the center and roll angle of the images are matched, the images are compared to each other using the shortest distance transform map. The star coordinates of the database stars are the input to the 2D lookup map created by the distance transform. For each star of the database, the distance to the closest star of the camera image is retrieved. These distances are stored in an array and are sorted on shortest distance. Figure 3.7 shows how the distance is calculated.

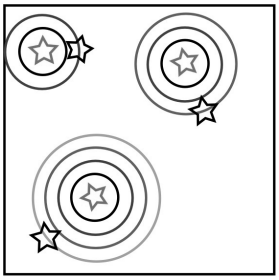


Figure 3.7: Distance calculation in the star tracker algorithm.

Assuming the distance between the concentric circles is one unit, the database star in the top left corner is 2 units of distance separated from its closest camera image star. The star in the top right corner has its closest camera image star at a distance of three units and the other star is four units of distance away. This results in the distance array (table 3.2). Based on this array, the algorithm can decide how well the images are alike.

$$\begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

Table 3.2: The distance array of figure 3.7.

Several decision criteria can be used to determine how well the images resemble each other based on this distance array. In this algorithm, a combination of two criteria is used. The combination of both gives better results than when the two criteria are used separately. The first criterion computes the total distance of a certain number of stars, the second criterion counts the number of database stars that are within a certain threshold distance of a camera star.

Distance criterion

This criterion sums up the distances of a reduced number of stars. This reduced number of stars is selected by taking the minimum of the number of database image stars and the number of camera image stars. When this number - N_{tot} - is obtained, the N_{tot} closest distances are added together. The smaller the total distance, the more the images are alike.

Close stars criterion

The number of distances that are smaller than a threshold value are counted and this is used as a criterion. Equal images will obviously have a lot of 'close stars', so the higher the number, the more the images are alike.
An important asset of this algorithm is that it can very accurately determine whether the offered solution is the correct one. Therefore the percentage of close stars is determined. The number of close stars, divided by N_{tot}, offers a very good value measure of the proposed solution. The power of this criterion will be presented in the simulations that are discussed in the next section.

3.2.5 Improvements in computational time

The calculation time for the algorithm can be drastically reduced by performing some minor modifications on the algorithm. These changes do not affect the robustness and were therefore implemented in the algorithm. The calculation time was first reduced by using the information obtained in the matching step to discard a lot of images that will not give a good comparison, before the actual comparison step takes place. A second reduction cuts the calculation time down significantly by making use of a preprocessed database. Both methods will be discussed next.

Discarding false images

A serious reduction in computation time can be obtained when part of the database images can be discarded before the actual comparison step. Information obtained in the matching step can be used to discard the vast majority of images without risking to lose the correct database image. The three angles and three distances between the centroid and the brightest stars offer a very good criterion to discard images. When two images are alike, these features will be approximately the same, otherwise they would not be matched correctly. The database images that have their six features within a predetermined margin of the camera image features are passed on to the comparison step. This margin is taken considerably large so that the robustness of the algorithm is not compromised. Because of the large variation in these features over the entire sky, around 90% of the images can be discarded.

Preprocessed database

Because the database images and the calculation of the centroid and angles in the matching step of the database images are always the same, the database can also be preprocessed to hold these values. For each database image, the position and magnitude of the stars in the image can be stored, together with all the possible combinations of centroids and smallest angles for the matching step. This eliminates a lot of computations and significantly enhances the speed of the algorithm. The preprocessed database only slightly increases memory usage.

3.2.6 Schematic overview

As a conclusion to the discussion of the algorithm, we present a schematic overview of the Shortest Distance Transform algorithm in Algorithm 1.

Algorithm 1 Outline of the Shortest Distance Transform Algorithm.

(1) Shortest distance transform (Star Positions, Magnitudes)
(2) Load database images
for $dataindex = 1 \rightarrow numberOfDatabaseImages$ **do**
  (3) Select database image (dataindex)
  (4) Match camera image and database image
  if Match is positive **then**
    (5) Input database information in Shortest Distance map
  end if
end for
(6) Withhold best image and return corresponding attitude

3.3 Test results

The performance of the algorithm was determined during a number of simulations with a variety of different noise conditions. Camera images are generated from the Hipparcos catalog and perturbations are added. Three kinds of perturbations were added to the images: perturbations on star positions, stars leaving the image and false stars entering the image. When the effect of stars leaving the image was tested, the brightest stars were left out to simulate a worst case scenario.
The camera was assumed to have a field of view of 20x20 degrees and a pixel array of 512x512. The minimum sensitivity of the camera was set at a visual stellar magnitude of 6. The calculations were performed on a Macbook Pro with 2.33 GHz Intel Core 2 Duo processor and the algorithm is written in Visual

C++ 6.
In each test session, 10,000 images generated uniformly over the database served as an input for the algorithm. For each image, the determined attitude is compared to the real attitude to verify the correctness of the algorithm. The percentage of close stars is also logged since this can serve as a criterion to determine the correctness of the determination. The calculation time is determined with the C++ command GetTickCount () and was also logged.

3.3.1 Positional error

The sensor information needed by the lost-in-space algorithm is given by the star acquisition algorithm. This algorithm processes the image taken by the camera and returns the position of the stars in the image and a measure for their magnitude. This positional information is subject to some noise. In the state of the art star acquisition algorithms the positional error amounts to around one tenth of a pixel [98]. Other sources of positional error are flaws in the image sensor or in the optics of the system. Especially in low cost projects, the lower quality of the components can generate a significant positional error. In order to test the effect of this positional error, Gaussian noise was added to the position of the stars. The mean was set to zero and tests were performed with increasing standard deviation to test the performance of the algorithm under increasing positional error. The standard deviation is expressed in arc seconds of deviation (figure 3.8).

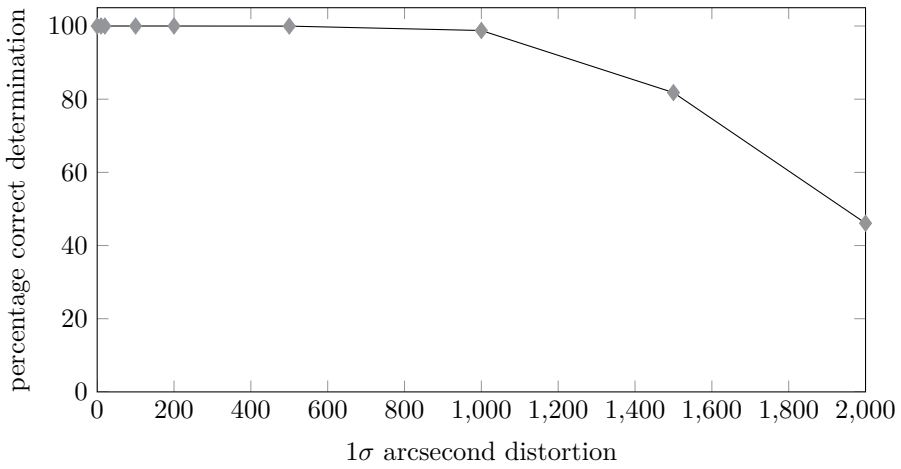


Figure 3.8: Effect of positional error on correct attitude determination.

The algorithm determines the attitude flawlessly up until a deviation of 200 arc seconds (1 σ) on the star positions. With positional errors of 1000 arc seconds (1 σ), the algorithm still determines almost 99 percent of the images correctly. In the tested setup, an error of 1000 arc seconds corresponds to a star being viewed more than seven pixels away from its true position. Such high positional error could be due to the effect of a rolling shutter and high slew rates or due to high optical distortions. Stars with high positional error might be discarded by the tracking algorithm that follows the lost-in-space algorithm because they reduce accuracy.
The comparison with other algorithms is very favorable for the Shortest Distance algorithm. The Pyramid Star Identification Technique [89], of which the authors say that it is extremely robust, determines 95.8 percent of images correctly when a deviation of around 4 arc seconds σ and a maximum of 24 false stars are present. The Oriented Triangles method [102], which was also designed to be very robust, recognizes 57 percent of the input stars when the positional error is 150 arc seconds σ.
The very significant lead of the Shortest Distance algorithm is obtained because this algorithm has a different approach than the existing algorithms. It does not use a combination of sets of stars to determine the position but uses all the stars in the image to compare the image to the database.

3.3.2 Missing stars

Due to the change in magnitude of stars, or due to malfunctioning pixels in the camera, sometimes a star tracker fails to notice a star. This effect was simulated in the test by discarding the brightest stars in the image. Because the matching step of the algorithm uses the brightest stars to match the images, a loss of these stars has the greatest negative effect on the performance of the algorithm. Simulations were done up to a loss of the six brightest stars in the image. This is shown in figure 3.9.
A small fraction (0.2 %) of images can not be determined when the brightest star is missing. When the three brightest stars are not captured by the camera, 90 percent of the images is still correctly determined. The odds that the brightest stars are missing in the camera image are really slim and the algorithm yields good results up to a loss of the three brightest stars, so it can be concluded that the robustness to this effect is also very high.

3.3.3 False stars

False stars are stars that are found in the camera image by the star detection algorithm but do not have a corresponding star in the database. Depending on what caused them, these false stars can have varying magnitudes. A number of causes that can generate false stars are given below:

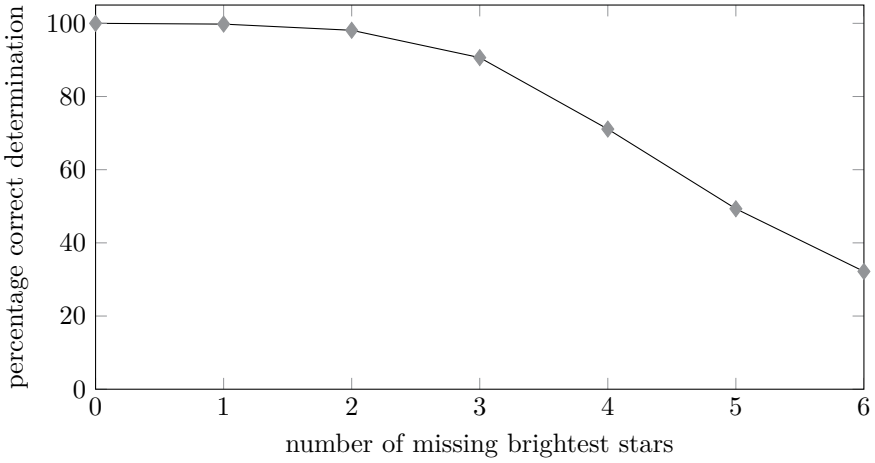


Figure 3.9: Effect of a loss of brightest stars on correct attitude determination.

- The camera could detect stars which have a higher magnitude than the limiting magnitude of the database. In this case, the detected star does in fact correspond to a real star, but this star has no corresponding database star and will be considered as a false star by the algorithm. The magnitude of these stars varies around the detection threshold of the camera and will therefore have higher magnitudes than the other detected stars.

- The various noise sources (shot noise, read noise, quantization noise,...) in the camera can generate spikes in the image data and this way generate false stars. For a quality detector, this effect is usually limited and false stars generated because of this will be faint [66].

- A man-made object (artificial satellite or space debris) passing in the field of view of the camera can reflect light of the sun on the star tracker camera and this way generate a false star. The probability that a man-made object passes the field of view was determined using the procedure of [109] for a star tracker with a field of view of $20° \times 20°$ and integration time of 0.1s in Low Earth Orbit. The resulting probability of having a man-made object in the field of view is around 1.44%, taking into account that the U.S. Space Surveillance Network had 16,094 cataloged objects approximately 10 cm or larger in July 2011 [67]. While most of the objects passing in the field of view will not be large enough to generate a false star or will only generate a faint false star, a small fraction of them, such as the International Space Station, may cause false stars which are brighter than all true stars in the image.

- A planet of our solar system in the field of view of the camera can generate a false star. The probability of viewing a planet rises when the field of view increases. Five planets can create false stars with stellar magnitude lower than that of the brightest true stars: Venus, Mars, Jupiter, Mercury and Saturn [96]. These can however be filtered out based on magnitude information in the star detection algorithm [110].

- Cosmic rays can generate bright false stars in the star tracker image. The probability of cosmic rays generating false stars is highly dependent on the environment of the satellite. However, bright signals generated by cosmic ray events can be distinguished from real stars if the stars are sufficiently defocussed. In that case, there is a considerable difference between the expected PSF of a star and the signal of a cosmic ray hit. Having a more defocussed star does reduce the signal to noise ratio for the centroiding algorithm. An additional approach is to take multiple consecutive frames and compare them, as there is very little chance of a cosmic ray event hitting the same pixel twice in quick succession [84]. This comes at the cost of more memory, CPU power and a longer computational time.

The effect of false stars was determined in two simulations. In the first simulation, the added false stars had magnitudes which were higher than those of the three brightest true stars. As is discussed above, false stars will in most cases have a larger magnitude than that of the brightest true stars. By choosing the magnitude of the false stars higher than that of the third brightest star, the false stars will not interfere with the matching procedure. In the second simulation, bright false stars with an apparent magnitude lower (equal to -2) than that of the brightest true stars were added. As mentioned above, there are cases in which false stars can be brighter than the brightest true stars. In this case, the false stars will interfere with the matching procedure. The effect of false stars and bright false stars on the correct determination of the attitude is given in figures 3.10 and 3.11.

The algorithm determines the attitude correctly with up to 400 false stars added to the image. When there are 650 false stars in the image, it still determines more than 98 percent of the images correctly. With an average amount of around 20 real stars in the image, we can clearly say that the Shortest Distance algorithm yields good results, even with a vast majority of false stars. When this is compared to the robustness of the Oriented Triangles algorithm, where 92 percent of the images is found with 3 false stars and 50 arc seconds of positional noise, it is clear that the shortest distance algorithm is very robust to false stars. This makes the Shortest Distance algorithm useful for star trackers that operate in highly hostile environments.

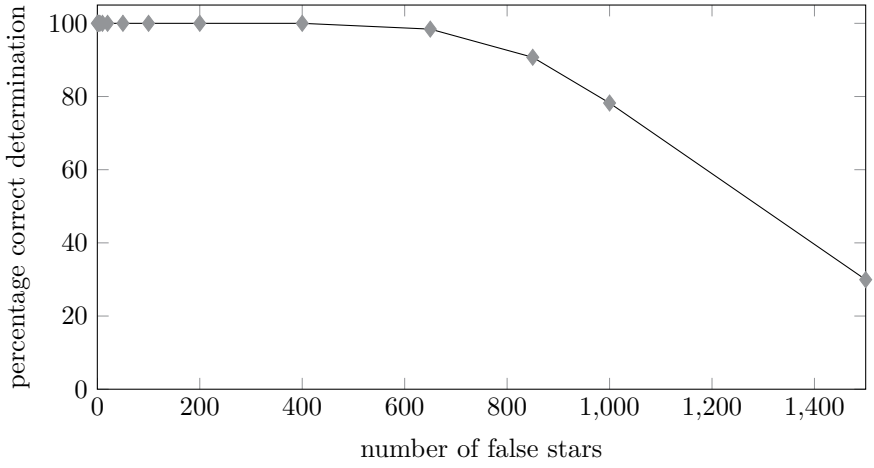


Figure 3.10: Effect of false stars on correct attitude determination.

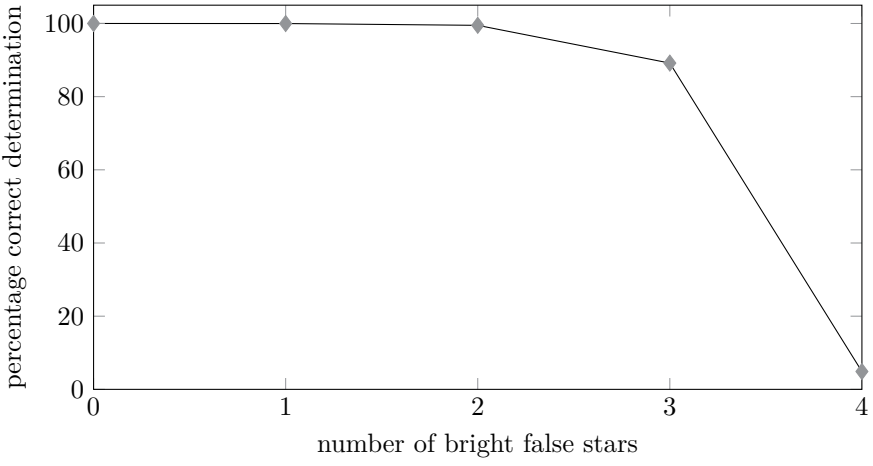


Figure 3.11: Effect of bright false stars on correct attitude determination.

When a false star with a higher brightness than that of all true stars is present in the image, only a small fraction (0.02%) of the images can not be determined. When three false stars with a magnitude lower than that of all true stars are added, around 90% of the images is still determined correctly. The odds that there are three false stars that are brighter than all true stars, are dependent on the environment of the satellite, the duration of acquisition and the detector. If many bright false stars are expected, the number of combinations in the

matching step can be increased, to allow more robustness to this effect.

3.3.4 Execution time and memory usage

The execution time of the algorithm is determined by:

- *The number of pixels of the camera image*: The execution time of the Shortest Distance procedure increases linearly with the number of pixels.
- *The number of combinations of brightest stars used in the matching step*: More combinations used in the matching step will increase the execution time of the matching step and will increase the number of comparison steps (As shown in table 3.1).
- *The number of database images*: The execution time of the algorithm increases linearly with the number of database images, as is depicted in figure 3.12.
- *The settings of the prefilter*: Stricter margins in the prefilter step will discard more images before the comparison step is executed, resulting in a lower execution time, as discussed in 3.2.5.
- *Preprocessing the database:* This reduces the number of calculations which need to be done and increases the speed of the algorithm as discussed in 3.2.5.

The results that were presented in this section were obtained with camera images of 512×512 pixels, making combinations of three out of the six brightest stars and using 1337 database images. While it can be seen on figure 3.12 that when using more than 400 database images, there are diminishing returns, a design choice has to be made between a faster algorithm with a smaller success rate and a somewhat slower algorithm with high robustness. Since robustness is key in the lost-in-space algorithm, we chose to use a large amount, i.e. 1337, of points for the tests. This leads to an image overlap of 97.43% as can be seen from equation 3.6.
Using these parameters, the execution time, averaged over 10 000 executions, was 75 milliseconds (see figure 3.12). The execution times were measured with the C++ command GetTickCount (). The memory required to store the preprocessed database (as discussed in section 3.2.5) is 4.7 Mb in this case.

$$O = \frac{(\frac{\gamma}{180^\circ}\pi)^2 - \frac{4\pi}{N_p}}{(\frac{\gamma}{180^\circ}\pi)^2} \times 100 \tag{3.6}$$

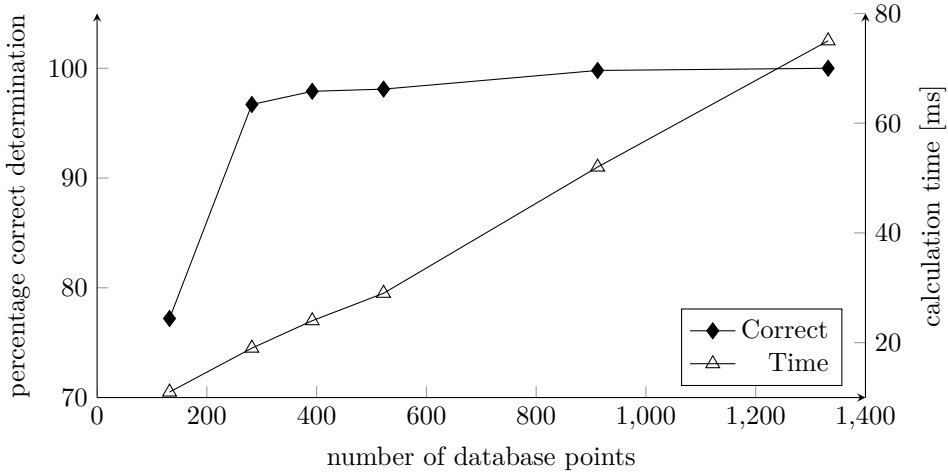


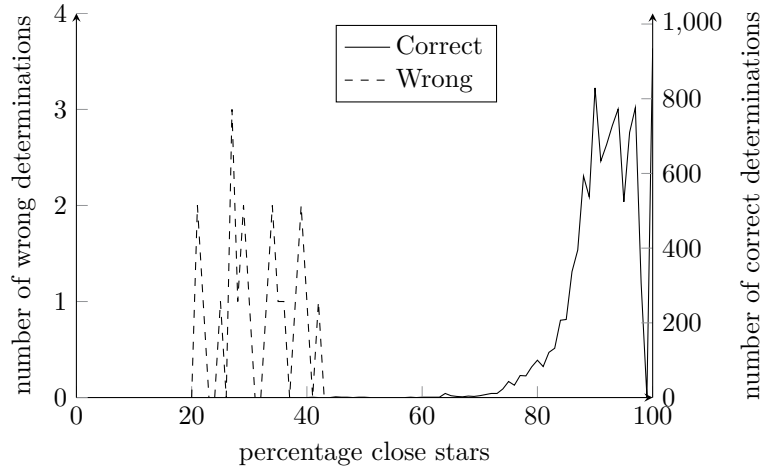
Figure 3.12: Calculation time and correct determinations versus number of database images.

3.3.5 Validation criterion

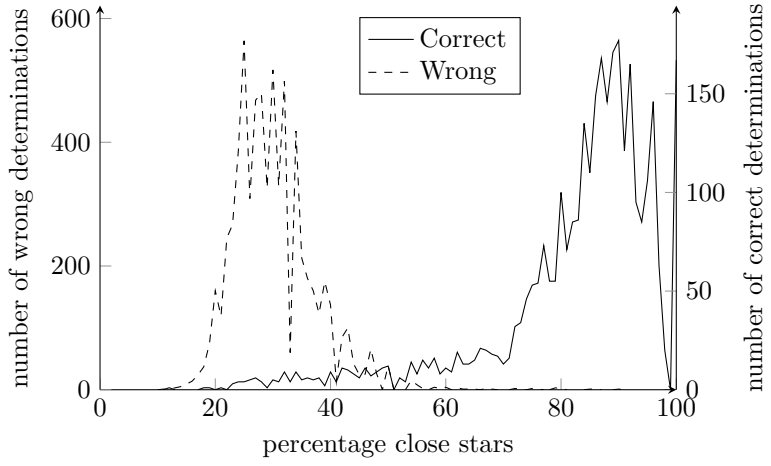
The algorithm will always return an attitude, even when it has not found the correct attitude. To be able to determine whether the output given by the algorithm is correct, we can use a validation criterion which proved to be very reliable during testing: the 'percentage of close stars'. When the Shortest Distance algorithm returns a false image, the percentage of close stars is generally low. In case the algorithm should not return any image, the percentage of close stars is zero. When the validation criterion is lower than a selected threshold value, the lost-in-space algorithm is repeated with a new camera image.

Graphical representation

To graphically show how the criterion separates the good determinations from the false determinations, we have drawn histograms of the criterion during tests with missing stars. The criterion is depicted on the horizontal axis and the two vertical axes hold the number of correct and false proposed solutions. The case where stars were missing was chosen because it was seen in tests that the algorithm is most sensitive to bright stars that are missing in the image. As can be seen in figure (3.13), the correct and wrong determinations are clearly separated by the criterion. Where a correct determination has a percentage of close stars of around 90%, a wrong determination leads to a percentage of close stars of around 30%. It can be seen that the criterion is very reliable in this case.



(a) 1 missing star



(b) 6 missing stars

Figure 3.13: The separation of correct and wrong determinations by the criterion (a) 1 missing star (b) 6 missing stars.

In the case of false stars (figure 3.14) and distortion on star positions (figure 3.15), the criterion also separates the correct and wrong determinations, albeit not as clearly as is the case with the missing stars. Given the fact that the algorithm is very insensitive to false stars and distortions, this is not a big issue.

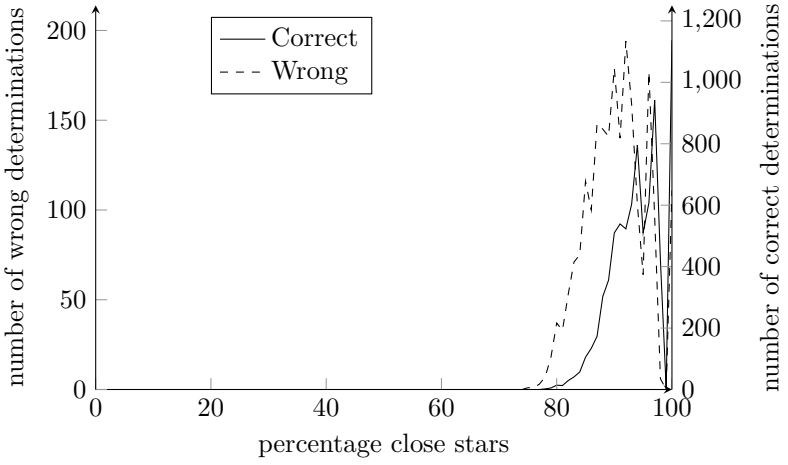


Figure 3.14: The separation of correct and wrong determinations with 1000 false stars.

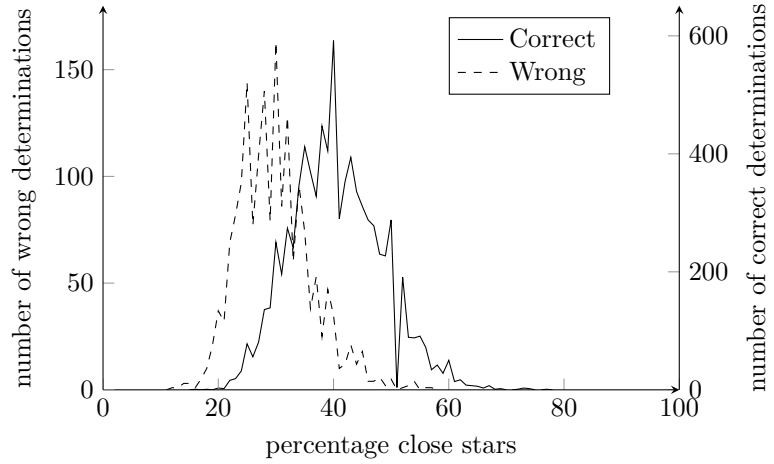


Figure 3.15: The separation of correct and wrong determinations with a positional error of 1500 arc seconds 1σ.

Implementation

The decision criterion was implemented with a rejection threshold of 50%. When the percentage of close stars is lower than this value, the attitude will be considered to be wrong and the lost-in-space algorithm is repeated with a new

camera image.
In the first case, a number of brightest stars are missing from the image. In table (3.3), the separation of the results in four categories is depicted. The results are separated based on whether they were correctly (C) or wrongly (W) determined and whether the decision criterion rejected (R) or passed (P) the result. It can be seen from this table that the vast majority of wrong attitudes is rejected by the criterion. Furthermore, the algorithm rejects very little correct attitudes, limiting the chance of an unnecessary repetition of the lost-in-space algorithm.

Table 3.3: Decision criterion separation in case of missing stars.

Number of missing stars	C/P	C/R	W/P	W/R
1	99.73	0.06	0	0.21
2	97.86	0.24	0.06	1.84
3	89.54	1.09	0.15	9.22
4	69.33	1.77	0.18	28.72
5	47.30	2.02	0.34	50.34
6	30.35	1.84	0.58	67.23

The separation into the four categories in the case of distortion on the pixel position is shown in table 3.4. The decision criterion again rejects almost all the wrong results. When the distortion on the pixel position increases, the number of rejected correct measurements increases too.

Table 3.4: Decision criterion separation in case of positional error.

Arc seconds (1 σ) distortion	C/P	C/R	W/P	W/R
100	100	0	0	0
500	99.85	0.12	0	0.03
1000	78.66	20.11	0.02	1.21
1500	10.49	71.31	0.14	18.06
2000	0.25	45.85	0.02	53.88

The wrong attitude determinations because of false stars with a magnitude higher than that of the third brightest real star are not rejected by the decision criterion. The algorithm only starts to fail when more than 400 false stars are in the image. With such a high number of false stars, the percentage of

close stars is always higher than the threshold and the criterion therefore does not reject attitude determinations, as can be seen in table 3.5. The decision criterion has no effect in this case.

Table 3.5: Decision criterion separation in case of false stars.

Number of false stars	C/P	C/R	W/P	W/R
100	100	0	0	0
300	100	0	0	0
500	98.37	0	1.63	0
800	87.42	0	12.58	0
1200	54.30	0	45.70	0

In the case where there are false stars with a lower magnitude than that of all true stars, the decision criterion rejects the vast majority of wrong results. The separation into the four categories can be seen in table 3.6.

Table 3.6: Decision criterion separation in case of bright false stars.

Number of bright false stars	C/P	C/R	W/P	W/R
1	99.95	0.03	0.01	0.01
2	99.25	0.24	0.11	0.40
3	86.74	2.43	0.97	9.86
4	3.37	1.51	6.36	88.76

In table 3.7, the results of the Shortest Distance algorithm and its decision criterion are shown for a selection of combined distortions on the camera image. The results in this table reflect the high robustness of the Shortest Distance algorithm, even when there is a combination of distortions in the image. It can also be seen that the decision criterion successfully separates the correct and wrong images, if the number of false stars is limited. Up from around 50 false stars, the decision criterion lets more false determinations pass than it rejects.

Table 3.7: Decision criterion separation in case of a combination of distortions.

# Missing stars	Arc sec. (1 σ) distortion	# Bright false stars	# false stars	C/P	C/R	W/P	W/R
1	100	0	20	99.48	0.09	0.17	0.26
1	100	0	50	99.47	0.04	0.29	0.20
1	100	1	50	99.32	0.10	0.25	0.33
1	400	0	100	98.20	0.14	1.24	0.42
1	400	1	100	97.18	0.24	1.91	0.67
1	400	1	300	93.45	0	6.55	0
2	400	1	300	83.06	0	16.94	0

3.4 Conclusion

The lost-in-space algorithm we presented here is based on the Shortest Distance Transform technique. This new approach to solve the lost-in-space problem leads to a fast algorithm which is a lot more robust to distortions in the camera image than existing algorithms. The algorithm proved in tests to be robust to missing stars, false stars and distortions on star positions. Furthermore, this algorithm has a reliable criterion to determine whether the attitude has been correctly determined.

A higher robustness of the lost-in-space algorithm allows the use of star trackers in hostile environments. Furthermore, the increased robustness on a software level reduces the cost to improve robustness on a hardware level, such as the cost to make the star tracker image sensor radiation hardened. A non-radiation hardened sensor will deteriorate more towards his end of life, which will increase noise and distortions. A more robust lost-in-space algorithm could allow to still initialize the star tracker sequence correctly with a more deteriorated sensor. This will allow smaller satellite missions with a lower budget to benefit from the pointing accuracy of the star tracker. This algorithm can thus extend the range of missions in which a star tracker can be used and will help make space missions more affordable for smaller institutions.

Chapter 4

Highly Efficient Attitude Estimation Algorithm for Star Trackers Using Optimal Image Matching

The contents of this chapter were published in the Journal of Guidance, Control and Dynamics: Delabie, T., De Schutter, J., Vandenbussche, B. (2013). Highly Efficient Attitude Estimation Algorithm for Star Trackers Using Optimal Image Matching. *Journal of Guidance, Control and Dynamics*, 32 (6), 1672-1680.

This chapter presents a novel attitude estimation algorithm for spacecraft using a star tracker. The algorithm is based on an efficient approach to match the stars of two images optimally on top of each other, hence the name of the algorithm: AIM (Attitude estimation using Image Matching). In tests, AIM proved to be as robust as the most robust existing methods, and faster than the fast iterative methods. On top of this, AIM allows us in a lot of cases to eliminate a computationally intensive coordinate conversion which normally precedes the attitude estimation algorithm. The computational cost of this conversion step is several times higher than that of the attitude estimation algorithm itself, so this elimination yields a huge increase in efficiency as compared to the existing algorithms.

4.1 Introduction

In many spacecraft missions, accurate knowledge of the orientation of the spacecraft in space is crucial. Examples are space telescopes observing an astronomical target or communication satellites that need to accurately point an antenna to a ground station. Several sensors to determine this orientation, also referred to as the attitude of the satellite, have been developed. The most accurate of these sensors is the star tracker. This sensor takes an image of the surrounding star field and compares it to a database of known star positions. Typically, this sensor can determine the attitude of the satellite with an accuracy in the range of a few arc seconds [115] or even sub arcsecond [95].
An autonomous star tracker operates in two different modes [75]. The first mode is the initial attitude acquisition. In this mode, the star tracker determines the attitude of the satellite without a priori knowledge [120, 26]. Once an initial attitude has been acquired, the star tracker switches to the tracking mode. In this second mode, where the star tracker has a priori knowledge, the database search can be limited, allowing fast and accurate attitude estimation.
A variety of attitude estimation algorithms that can be used in this tracking mode have been proposed. All of these estimate the spacecraft attitude from vector measurements and seek the matrix A that minimizes the loss function proposed by Wahba [79]:

$$L(A) = \frac{1}{2}\sum_{i=1}^{n} w_i |\mathbf{b}_i - A\mathbf{r}_i|^2, \tag{4.1}$$

where $\mathbf{b}_i$ are the unit vectors observed in the spacecraft body frame, $\mathbf{r}_i$ are the corresponding unit vectors in a reference frame and w_i are non-negative weights.

The most robust of these attitude estimation algorithms are Davenport's q method [77] and the Singular Value Decomposition method [71]. These methods are slow, but are based on robust algorithms to calculate the symmetric eigenvalue problem and the SVD [60, 51]. Fast iterative solutions to Wahba's problem have been developed [17]. These solutions solve the characteristic equation for the maximum eigenvalue and use this value to construct the optimal attitude quaternion. This method was first used in QUEST [111, 114], which is still the most widely used algorithm to solve Wahba's problem. QUEST was followed by FOAM [78], ESOQ [86] and ESOQ2 [87]. The main goal of the algorithms following QUEST was to improve the computational efficiency. However, the improvements are small, if any, so these algorithms could be considered to be equally fast [17].
These attitude estimation algorithms all share one important drawback, when used with a star tracker. They determine the attitude of the spacecraft based on observations which are represented as unit vectors. In a star tracker however,

the observations are 2D star centroids on the focal plane. The conversion of these 2D coordinates to unit vectors is not straightforward. Because of optical and electronic distortions, temperature, magnetic and star intensity effects, an empirical model based on laboratory calibrations is often used to convert the coordinates [42].
In this chapter, an algorithm is proposed which is faster than the algorithms mentioned above, and more importantly, in some cases eliminates the need for the computationally intensive coordinate conversion during each attitude determination step. This way, the computational cost of the entire attitude estimation procedure is highly reduced. This increased efficiency allows us to obtain the attitude information at a higher rate or track more stars, improving the accuracy of the attitude estimation. Furthermore, in the growing market of small satellites, this method could reduce the computational expense of the spacecraft bus, allowing to carry a more demanding payload.
In section 4.2, the algorithm, referred to as AIM (Attitude estimation using Image Matching), is presented. Section 4.3 discusses the accuracy, speed and robustness of AIM, compared to the state of the art algorithms. These comparisons show that the speed of AIM greatly exceeds that of existing algorithms, while obtaining similar accuracy and robustness provided by the robust estimators.

4.2 Algorithm

In this section, the AIM algorithm will be derived. Before discussing the attitude estimation algorithm itself, it is important to examine the preceding algorithms, which calculate the input for the actual attitude estimation algorithm. These algorithms require a number of calculations which is an order of magnitude higher than that of the attitude estimation algorithm itself. Therefore it is important to examine the influence of the attitude estimation algorithm on these preceding steps, in view of the comparison with other attitude estimation algorithms. This section starts with an overview of the algorithms that lead to the input of the attitude estimation algorithm (as seen in figure 4.1), before deriving the AIM algorithm itself.

Figure 4.1: The sequence of algorithms which are used to estimate the attitude of the spacecraft with a star tracker.

4.2.1 Centroiding

The first step in the sequence of algorithms is to determine the centroids of the observed stars in the camera image. Since the previous attitude and rate information is known in tracking mode, the position of stars in the image can be predicted, based on the star image coordinates of the previous image [75]. This prediction step eliminates the need to digitize the entire image, and instead allows to only digitize small windows at the predicted star positions. In those windows, a centroiding algorithm [98, 103] is used to accurately determine the centroids of the stars in the camera image.

4.2.2 Star identification

The identification of the image stars, i.e. determining to which database star each observed star corresponds, is referred to as star identification. Samaan presented two methods for star identification in [105]. In this paper, an adaptation of the star identification algorithm of [26] was used. This algorithm calculates the similarity between the camera image and preprocessed database images using the Shortest Distance Transform. The database image with the highest similarity is retained and the stars can be identified this way. This algorithm is significantly more robust than other star identification algorithms. In addition, it is fast and gives a very reliable performance value which indicates whether or not the stars have been correctly identified. In this adaptation, the database images used in reference [26] are sorted so that the images lying closest to the previous attitude are checked first. The algorithm is stopped when one of the database images yields a performance value above a predetermined threshold. This is usually already the case in the first image, which makes this approach extremely fast, on top of being very robust to distortions in the camera image.

4.2.3 Coordinate conversion

While the previous steps were the same for both AIM and the existing algorithms, in this step, there is an important difference. First, the classical approach is described, followed by the approach used by AIM.

Classical approach

In the current state of the art algorithms, the attitude is determined using unit vectors (i,j,k). Therefore, the next step is to convert the focal plane coordinates (x,y) of the observed stars to unit vectors. This can be done using a simple pinhole model as described in [42]:

$$\begin{cases} i = \frac{\frac{x-x_0}{F}}{\sqrt{1+(\frac{x-x_0}{F})^2+(\frac{y-y_0}{F})^2}} \\ j = \frac{\frac{y-y_0}{F}}{\sqrt{1+(\frac{x-x_0}{F})^2+(\frac{y-y_0}{F})^2}} \\ k = \frac{1}{\sqrt{1+(\frac{x-x_0}{F})^2+(\frac{y-y_0}{F})^2}} \end{cases} \tag{4.2}$$

In these equations, (x, y) is the coordinate of the star in the image, (x_0, y_0) is the intersection of the focal plane and the optical axis, (i, j, k) is the unit vector of the observed star, described in the sensor reference frame, and F is the focal length of the optical system.
To account for distortions however, the image plane coordinates are first corrected for distortions using a fifth order polynomial [42]:

$$x_c = -k_0 + k_1 x + k_2 y + k_3 x(x^2 + y^2) + k_4 x(x^2 + y^2)^2 - k_5 x^2 - k_6 xy - k_7 y^2 \tag{4.3}$$

$$y_c = -h_0 + h_1 y + h_2 x + h_3 y(y^2 + x^2) + h_4 y(y^2 + x^2)^2 - h_5 y^2 - h_6 yx - h_7 x^2 \tag{4.4}$$

Where (x_c, y_c) are the coordinates after they have been corrected for camera distortions, and the parameters (k_i, h_i) depend on the camera. The coordinates (x_c, y_c) are then converted using equations 4.2.

Approach of AIM

A first difference between the state of the art algorithms and AIM is that AIM determines the attitude using focal plane coordinates. While the existing algorithms convert the observed focal plane coordinates to unit vectors, AIM will convert the unit vectors of the database stars, which were selected in the star identification step, to focal plane coordinates. The fact that the conversion is done on the database stars (of which the coordinates do not change) yields a significant increase in efficiency. This is explained and quantified in section 4.3.1.
To convert the database unit vectors to coordinates in the focal plane, the selected database unit vectors first need to be rotated so that they are centered around the k-axis. In order to perform this rotation, a quaternion q_{dat} is selected. This quaternion is a coarse estimate of the current attitude, being the attitude where the unit vectors are observed. This leads to the second distinction between AIM and the state of the art algorithms: AIM requires a coarse estimate of the current attitude, in order to be able to estimate the current attitude, while the existing algorithms do not. In tracking mode, the attitude estimation obtained in the previous time step can be used as q_{dat}, since

the attitude changes between subsequent time steps are generally small. This is further discussed in section 4.3.2. In lost-in-space mode, the star identification algorithm of [26] was used. This algorithm identifies the stars in the FOV and also presents a coarse estimate of the attitude. This estimate is then used as q_{dat}.

In the first step of the coordinate conversion for AIM, the unit vectors are rotated over the inverse of q_{dat} to center them around the k-axis. Since q_{dat} is a unit quaternion, the inverse is equal to its conjugate. To rotate the database unit vector $\mathbf{v} = (\hat{i}, \hat{j}, \hat{k})$ over the inverse of the quaternion q_{dat}, equation 4.5 is used.

$$\mathbf{v}_r = \bar{q}_{dat}\mathbf{v}q_{dat} \tag{4.5}$$

In this equation, $\mathbf{v} = (\hat{i}, \hat{j}, \hat{k})$ is the unit vector of a star as found in the database, $\mathbf{v}_r = (\hat{i}_r, \hat{j}_r, \hat{k}_r)$ is that unit vector after rotation by the inverse of q_{dat}, and a bar over a quaternion indicates that the conjugate of the quaternion is taken. After this, the conversion from unit vector coordinates to database focal plane coordinates $(\hat{x}, \hat{y})$ is performed. Using the pinhole model, this conversion is performed with the following equations:

$$\begin{cases} \hat{x} = x_0 + F\hat{i}_r/\hat{k}_r \\ \hat{y} = y_0 + F\hat{j}_r/\hat{k}_r \end{cases} \tag{4.6}$$

The resulting coordinates $(\hat{x}, \hat{y})$ are the focal plane coordinates an ideal star tracker would observe if its attitude was q_{dat}. Because a real star tracker has camera distortions, these coordinates are transformed to account for these distortions. Using the inverse transformation of equations 4.3 - 4.4, coordinates $(\hat{x}, \hat{y})$ are transformed to the values as they would be measured with the distortions in the camera. The polynomials to calculate this transformation are given in equations 4.7 and 4.8.

$$\begin{aligned} \hat{x}_d = &- k_{0in} + k_{1in}\hat{x} + k_{2in}\hat{y} + k_{3in}\hat{x}(\hat{x}^2 + \hat{y}^2) + k_{4in}\hat{x}(\hat{x}^2 + \hat{y}^2)^2 \\ &- k_{5in}\hat{x}^2 - k_{6in}\hat{x}\hat{y} - k_{7in}\hat{y}^2 \end{aligned} \tag{4.7}$$

$$\begin{aligned} \hat{y}_d = &- h_{0in} + h_{1in}\hat{y} + h_{2in}\hat{x} + h_{3in}\hat{y}(\hat{y}^2 + \hat{x}^2) + h_{4in}\hat{y}(\hat{y}^2 + \hat{x}^2)^2 \\ &- h_{5in}\hat{y}^2 - h_{6in}\hat{y}\hat{x} - h_{7in}\hat{x}^2 \end{aligned} \tag{4.8}$$

In these equations, $(\hat{x}_d, \hat{y}_d)$ are the database star coordinates after the camera distortions have been added, and (k_{iin}, h_{iin}) are the parameters which are specific for the camera.

4.2.4 Attitude estimation algorithm

Once the preceding steps have been finished, the actual attitude estimation algorithm is executed. At the heart of AIM lies an efficient optimization to find the transformation that matches the stars of the camera star image and of the database star image optimally on top of each other. This transformation is then used to determine the estimated difference, q_{diff}, between the attitude at which the camera star image was taken (being the true attitude, q_t) and the attitude at which the database star image was selected, q_{dat}. Finally, the estimated attitude of the satellite, q_e is calculated from this.

Construction of the cost function

In order to find the transformation which optimally matches the stars of two images on top of each other, a cost function which needs to be minimized is constructed. This cost function is the sum of the euclidean distances squared between each pair of image star and their corresponding transformed database star. These stars were paired with each other in the star identification step. Each image star was identified and matched with a star in the database. If no match is found, the star is discarded so that the number of image stars and database stars in this cost function is always the same.
The distance is multiplied by a weight which is specific for each star pair. This weight could be determined based on e.g. a confidence measure for the centroid, calculated in the centroiding algorithm. This way, a star of which the centroid has been determined with higher confidence in the centroiding algorithm, could be given more value in the tracking step. The cost function is constructed as follows:

$$\Gamma_{AIM}(\phi, t_x, t_y) =$$

$$\sum_{i=1}^{n_s} w_i((x_i - \hat{x}_i cos(\phi) + \hat{y}_i sin(\phi) - t_x)^2 + (y_i - \hat{x}_i sin(\phi) - \hat{y}_i cos(\phi) - t_y)^2) \tag{4.9}$$

In this function, w_i is the weight given to star i, (x_i, y_i) are the coordinates of observed star i in the focal plane, $(\hat{x}_i, \hat{y}_i)$ are the coordinates of the corresponding database star i in the focal plane, ϕ is the angle over which the database stars are rotated with respect to the origin of the frame in which the database coordinates are described, t_x and t_y are the distances over which the database stars are translated in x- and y-direction respectively, and n_s is the number of stars used in the attitude estimation algorithm. This is depicted in figure 4.2. In this figure, the four stars of the camera image (x_i, y_i) are visible in a lighter shade and filled on the bottom left corner. The database stars $(\hat{x}_i, \hat{y}_i)$ are depicted in

black. On figure 4.2b the database stars are rotated over an angle ϕ. On figure 4.2c, they are translated over a distance t_x. Finally, they are translated over a distance t_y on figure 4.2d. The total distance between the camera stars and the transformed database stars as seen in 4.2d is the cost function. The database stars of the previous image are shown in dashed lines.

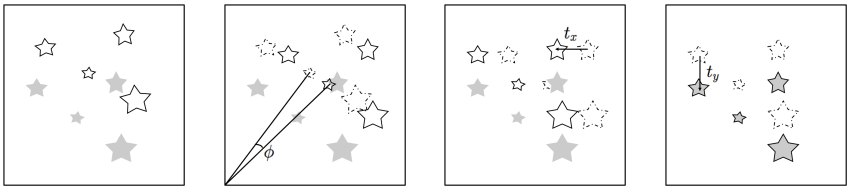


Figure 4.2: The database stars are transformed to minimize the distance with their corresponding camera stars.

Minimization of the cost function

To achieve an optimal matching of the transformed database stars with the camera image stars, the total distance between the corresponding stars needs to be minimized:

$$min(\Gamma_{AIM}(\phi, t_x, t_y)) \tag{4.10}$$

The unknown variables ϕ, t_x, and t_y which minimize this cost function can be found by calculating the derivative of this cost function to each of the unknowns, setting the three obtained equations equal to zero and solving this system of three equations.

$$\begin{cases} \frac{d\Gamma_{AIM}}{d\phi} = 0 \\ \frac{d\Gamma_{AIM}}{dt_x} = 0 \\ \frac{d\Gamma_{AIM}}{dt_y} = 0 \end{cases} \tag{4.11}$$

Generally, solving this system of three equations can take a significant amount of work and lead to very involved expressions, but this is not the case for the AIM algorithm. By substituting the second and third equation into the first, the three transformation variables which optimally map the database image stars on top of the camera image stars can be explicitly calculated immediately. This is an extremely fast procedure. Moreover, because no equation solving or complex calculations are used, this procedure is very robust. These three explicit relationships are given in the following equations:

$$\phi = atan2(sx.s\hat{y} - sy.s\hat{x} - sx\hat{y} + sy\hat{x}, -sx.s\hat{x} - sy.s\hat{y} + sx\hat{x} + sy\hat{y}) \quad (4.12)$$

$$t_x = (sx - \frac{b}{2}).cos(\phi) + (sy - \frac{b}{2}).sin(\phi) + \frac{b}{2} - s\hat{x} \quad (4.13)$$

$$t_y = -(sx - \frac{l}{2}).sin(\phi) + (sy - \frac{l}{2}).cos(\phi) + \frac{l}{2} - s\hat{y} \quad (4.14)$$

Here, b and l are the maximum values the pixel coordinates can have in the focal plane, $(\frac{b}{2}, \frac{l}{2})$ is therefore the coordinate of the center of the focal plane. The other variables in these three equations are calculated as follows:

$$\begin{aligned} sx &= \sum_{n=1}^{n_s} w_i.x_i & sx\hat{x} &= \sum_{n=1}^{n_s} w_i.x_i.\hat{x}_i \\ sy &= \sum_{n=1}^{n_s} w_i.y_i & sx\hat{y} &= \sum_{n=1}^{n_s} w_i.x_i.\hat{y}_i \\ s\hat{x} &= \sum_{n=1}^{n_s} w_i.\hat{x}_i & sy\hat{x} &= \sum_{n=1}^{n_s} w_i.y_i.\hat{x}_i \\ s\hat{y} &= \sum_{n=1}^{n_s} w_i.\hat{y}_i & sy\hat{y} &= \sum_{n=1}^{n_s} w_i.y_i.\hat{y}_i \end{aligned} \quad (4.15)$$

Calculation of the attitude quaternion

The three transformation values, which are readily calculated from equations 4.12 - 4.14, are then converted to three Euler angles:

$$\phi = \phi \quad (4.16)$$

$$\theta = atan(\gamma.\frac{t_y}{l}) \quad (4.17)$$

$$\psi = atan(\gamma.\frac{t_x}{b}) \quad (4.18)$$

where γ is the camera field of view. Because most calculations in the attitude determination and control system are done using quaternions, these Euler angles are then converted to a quaternion, the difference quaternion q_{diff}. To convert Euler angles to a quaternion, equation 4.19 is used:

$$q_{diff} = \begin{pmatrix} cos(\phi_h)cos(\theta_h)cos(\psi_h) + sin(\phi_h)sin(\theta_h)sin(\psi_h) \\ sin(\phi_h)cos(\theta_h)cos(\psi_h) - cos(\phi_h)sin(\theta_h)sin(\psi_h) \\ cos(\phi_h)sin(\theta_h)cos(\psi_h) + sin(\phi_h)cos(\theta_h)sin(\psi_h) \\ cos(\phi_h)cos(\theta_h)sin(\psi_h) - sin(\phi_h)sin(\theta_h)cos(\psi_h), \end{pmatrix} \tag{4.19}$$

where ϕ_h, θ_h and ψ_h are the half of the angles ϕ, θ and ψ, respectively. Since these angles are small, this conversion is simplified to equation 4.20:

$$q_{diff} = \begin{pmatrix} 1 + \phi_h\theta_h\psi_h \\ \phi_h - \theta_h\psi_h \\ \theta_h + \phi_h\psi_h \\ \psi_h - \phi_h\theta_h \end{pmatrix} \tag{4.20}$$

This quaternion represents the estimated rotation needed to rotate the database quaternion, q_{dat}, to the true attitude quaternion q_t. Finally, by multiplying q_{diff} with q_{dat}, the estimated quaternion q_e is obtained:

$$q_e = q_{diff} \otimes q_{dat} \tag{4.21}$$

Small-angle approximation

The three Euler angles calculated in formulas 4.16 - 4.18 represent the difference between the estimated satellite attitude and the attitude of the database image. This database image is taken at a previously estimated satellite attitude, q_{dat}. Since the updating frequency of a star tracker is generally between 1 and 10 Hz and since the rotational velocity of a satellite is generally under $1^\circ/s$, the difference between the estimated satellite attitude, q_e and the attitude of the database image, q_{dat} is generally smaller than 0.1° around each axis. Because the Euler angles are close to zero, the trigonometric functions in equations 4.12 - 4.14 and equations 4.17 - 4.18 were approximated by their Taylor series up to the first order:

$$sin(\theta) \approx \theta \tag{4.22}$$

$$cos(\theta) \approx 1 \tag{4.23}$$

$$atan(\theta) \approx \theta. \tag{4.24}$$

Up to an angle of 0.25°, the error remains under one thousandth of a percent, which is negligible for this application. Since trigonometric functions are computationally expensive, this approximation was implemented in the AIM algorithm.

4.3 Testing

The performance of AIM was determined during a series of tests which validated the speed, accuracy and robustness of AIM. These results are compared to the results of QUEST and the q-Davenport method, respectively one of the fast-iterative solutions and one of the robust solutions. These three algorithms were implemented in MATLAB and C++. The implementation of QUEST and the q-Davenport method is based on the formulas given in [79]. For the QUEST algorithm, the adapted version of [16] was used, because this solved some problems with robustness. Stars up to a magnitude of 6 are used in the calculations. The tests were performed on a Dell Latitude Laptop with a 2.60 GHz Intel Core i7 processor and 8GB RAM. The C++ code was executed in Microsoft Visual C++ 6, with the /O2 flag to maximize speed and the /ML library compiler option for static single-threaded libraries.

4.3.1 Speed

A faster attitude estimation algorithm presents the advantage of allowing the control system to achieve the attitude information at a higher rate [88]. For this reason, the algorithms following QUEST were developed to increase the speed. A good overview of the speed of the most common estimation algorithms can be found in reference [17].

Flop count

The amount of floating-point operations (flops) for each of the existing algorithms was determined in [17]. In table 4.1, the amount of floating point operations for AIM is added to those results. The amount of flops is a function of the number of stars, n_s, used in the tracking algorithm. For QUEST, it is also a function of the number of Newton-Raphson iterations. In the q-Davenport method, the symmetric eigenvalue problem is solved. This can be done with different algorithms, which are generally iterative in nature. Because of this, a straightforward result can not be given for q-Davenport and it is therefore omitted from table 4.1.
From table 4.1, we see that thanks to the very efficient procedure of AIM, the number of flops needed is lower than the flops needed by QUEST. When 25 stars are used, AIM reduces the number of flops with 25% compared to QUEST.

C++ execution times

The execution time of AIM, QUEST and q-Davenport was measured using the C++ timer QueryPerformanceCounter during simulations in C++. Simulations of 100 million estimations were run with a different number of stars in the image (n_s). For QUEST, simulations were run with a different number of

Table 4.1: Floating point operations for the different attitude estimation algorithms.

Method	AIM	QUEST		
Iterations		0	1	2
$n_s = 4$				
	125	188	225	236
$n_s = 9$				
	205	293	330	341
$n_s = 25$				
	461	629	666	677

Newton-Raphson iterations to calculate the largest eigenvalue. The calculation times per execution are given in microseconds in table 4.2.

Table 4.2: C++ execution times for the different attitude estimation algorithms (in μs).

Method	AIM	QUEST			q-Davenport
Iterations		0	1	2	
$n_s = 4$					
	0.13	0.20	0.23	0.24	5.38
$n_s = 9$					
	0.19	0.29	0.31	0.33	5.58
$n_s = 25$					
	0.36	0.57	0.58	0.60	5.67

AIM is significantly faster than both QUEST and q-Davenport. The computational time is around one third lower than that of the fast method QUEST and an order of magnitude lower than that of the robust method q-Davenport. The calculation time also rises slower when the number of stars increases. This is because AIM estimates the attitude of the satellite using 2-D coordinates in stead of 3-D coordinates. Another important reason for this higher speed is that AIM does not require computationally intensive operations (such as an eigenvalue calculation for q-Davenport).
Since the coordinate conversion step is different for AIM and the state of the art algorithms, it is also important to assess the speed of the algorithms with the coordinate conversion step included. This is quantified in table 4.3, which shows the speed results of AIM, QUEST and q-Davenport in microseconds, with the coordinate conversion step included. To convert the coordinates, a pinhole

model was used. For AIM this is equation 4.6, for QUEST and q-Davenport this is equation 4.2. Camera distortions were also taken into account in this step using the polynomials of equations 4.7 and 4.8 for AIM, and equations 4.3 and 4.4 for QUEST and q-Davenport. The parameters of these polynomials were chosen to be the same as the parameters used in the Herschel star tracker, in order to get a realistic result [42]. These parameters are given in table 4.4.

Table 4.3: C++ execution times for the different attitude estimation algorithms with the coordinate conversion step included (in μs).

Method	AIM	QUEST			q-Davenport
Iterations		0	1	2	
$n_s = 4$					
	0.36	0.52	0.54	0.56	5.74
$n_s = 9$					
	0.62	0.92	0.95	0.97	6.26
$n_s = 25$					
	1.48	2.25	2.27	2.28	7.42

Table 4.4: Parameters of the polynomials used in the camera distortion model.

i	k_i	h_i	$k_{i_{inv}}$	$h_{i_{inv}}$
0	-1.27866006 e-4	2.0675277 e-5	1.3025586 e-4	-1.8755991 e-5
1	1	1	0.99999929	0.99999960
2	8.57471666 e-5	-1.60933572 e-5	-8.5120363 e-5	1.6173263 e-5
3	2.19016482 e-4	2.14748994 e-4	2.1871748 e-4	2.1458371 e-4
4	-5.20164707 e-7	-2.26158663 e-7	6.3269083 e-7	3.4805732 e-7
5	-6.53773139 e-5	-4.54670784 e-5	6.4147788 e-5	4.4630672 e-5
6	-5.29575523 e-5	-4.89014428 e-5	5.2131004 e-5	4.8010159 e-5
7	-2.24629447 e-5	-2.27073830 e-5	2.2034513 e-5	2.2337249 e-5

The coordinate conversion step precedes the attitude estimation algorithm and is different for AIM and the state of the art methods. When the calculation time of this coordinate conversion is taken into account, it becomes clear from table 4.3 that the procedure of AIM is faster than that of QUEST and q-Davenport. The calculation time of the AIM procedure is around one third lower than that of QUEST. It can be concluded that AIM yields a large speed increase.
In some cases, AIM allows to eliminate the coordinate conversion step, this way speeding up the procedure greatly. This is discussed in the next section.

Elimination of a preceding algorithm

When the tracked stars remain the same for some period, the same transformed database stars can be used for several exposures. This means the unit vectors of the database stars can be converted to image coordinates one time, and be used for a number of following exposures. This in contrast to the current algorithms, where the camera star coordinates need to be converted to unit vectors every time. When the same database stars can be used for several exposures, the algorithm sequence of figure 4.1 changes for the AIM algorithm to the shorter algorithm sequence of figure 4.3. The database image then only needs to be updated when stars enter or leave the focal plane or when the observed stars and database stars have strayed too much from each other. In the next section it is validated when a database image should be updated in this case.

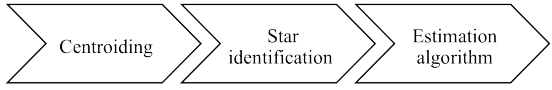


Figure 4.3: The shorter sequence of algorithms for the AIM algorithm when the same database stars can be used for a period of exposures. Eliminating the conversion of coordinates yields a significant reduction in calculation time.

The increased efficiency is especially important when the star tracker is in fine pointing mode. In this mode, the payload of the satellite needs to stay fixed on a certain point for a long time and the attitude of the satellite changes slowly. It is in this mode, which is very common for satellites, that AIM can very efficiently use the same database image for a very long period of time, this way totally eliminating a very computationally intensive calculation.
In table 4.5, the calculation times of AIM are compared to those of the state of the art algorithms, for the case where AIM can reuse its database image. Because of the relatively large computational expense of converting the coordinates in the state of the art approaches, the speed increase yielded by AIM is very high in this case. When 25 stars are tracked, the time needed to estimate the spacecraft attitude after star identification is more than 6 times lower for AIM than for the state of the art 'fast' method QUEST.

Table 4.5: C++ execution times for the different attitude estimation algorithms (in μs) when AIM can reuse its database image

Method	AIM	QUEST			q-Davenport
Iterations		0	1	2	
$n_s = 4$					
	0.13	0.52	0.54	0.56	5.74
$n_s = 9$					
	0.19	0.92	0.95	0.97	6.26
$n_s = 25$					
	0.36	2.25	2.27	2.28	7.42

4.3.2 Accuracy

The more accurate a star tracker is, the more precisely it allows a spacecraft to point the payloads to specific targets. This allows for better observations and more valuable data. A star tracker can typically determine the attitude of the spacecraft with an error of a few arcseconds or even sub arcsecond cross boresight, and an error which is typically 6-16 times larger around the roll axis [75].

Test Set-up

For each simulation of 10,000 star tracker exposures distributed randomly over the sky, values were chosen for the field of view γ, the accuracy of the centroiding E_{cent}, the number of pixels in one row of the camera n_{pix}, and the number of stars used in the estimation algorithm, n_s. These values will always be mentioned when the results are presented. To test the accuracy, noise was added to the camera image. This was done by adding Gaussian white noise with a variance equal to E_{cent} to the known correct pixel coordinates. In order to simulate camera distortions, the correct pixel coordinates were transformed using the polynomials of equations 4.7 - 4.8 with the parameters of table 4.4.

Projection distortions

The AIM algorithm optimally matches 2D star images on top of each other. A problem which arises here is that some distortion is always induced when a 3D image is projected onto a plane [40]. This means that the position of stars relative to each other is slightly different when the star image is taken with a different camera center. These distortions are larger if the difference between the centers is larger. The projection distortions decrease the accuracy when the difference between the attitude at which the camera image was taken, q_t, and the attitude at which the database stars were taken, q_{dat}, increases. In other words, the accuracy decreases when the coarse estimate of the current attitude,

q_{dat}, is further off from the true attitude q_t. It will be shown however that this decrease in accuracy is insignificant in a realistic environment.
To quantify this error, the increase in root mean square errors of AIM is shown with respect to the optimal q-Davenport solution as a function of the difference in attitude of database and image center in table 4.6.

Table 4.6: The percentage of the rms attitude angle errors increase of AIM compared to the q-Davenport error as a function of the difference in attitude at which the image was observed and the attitude at which the database stars were taken. $\gamma = 8°$, $n_{pix} = 1024$, $E_{cent} = 0.5$, leading to an rms error of around 4.9 arc seconds in cross boresight and around 90 arc seconds around the roll axis.

attitude difference α_r (arc seconds)	0	10	100	200	500	700
ψ error increase (%)	0	0	0	0.19	1.33	4.64
θ error increase (%)	0	0	0	−0.11	0.56	2.97
ϕ error increase (%)	0	0	0.01	0.06	0.09	0.18

The attitude difference value, α_r, represents the difference in arc seconds between the Euler angles ψ, θ and ϕ of the attitude at which the image was taken, q_a, and the attitude at which the database stars were selected, q_{dat}. The three bottom rows show the percentage with which the rms attitude angle errors of AIM are higher than the q-Davenport error for the three attitude angles (the first two are cross boresight and the last row is roll). Up to a difference of 100 arc seconds, the rms error for both methods is almost identical. This means that AIM and q-Davenport output an equally accurate estimate of the attitude, when the difference in Euler angles of the actual attitude and the attitude at which the database was selected is lower than 100 arc seconds. With a difference in Euler attitude angles of 500 arc seconds, AIM yields an error in cross boresight which is around 1 percent higher than that of q-Davenport. The error around the roll axis only slightly increases.
To validate the practical relevance of this effect, we note that during normal operation of the star tracker, the database stars are selected around the previous attitude. The difference between the attitude at which the star image was observed (the current spacecraft attitude) and the attitude at which the database was taken (the previous spacecraft attitude), depends on the rotational speed of the spacecraft and the frequency at which images are taken.
As a real life example, from the technical specifications of the CT-602 Star Tracker of Ball Aerospace [9], it can be seen that the star tracker offers full performance up to a tracking rate of 0.3 deg/sec and reduced performance up to 1.5 deg/sec, while having an update rate of 10 Hz. Therefore, the difference in

attitude between two consecutive exposures can for this star tracker maximally be ($0.3 \frac{^\circ}{sec}$. $3600 \frac{arcsec}{^\circ}$ / $10 \frac{1}{s}$ =) 108 arc seconds with full performance and ($1.5 \frac{^\circ}{sec}$. $3600 \frac{arcsec}{^\circ}$ / $10 \frac{1}{s}$ =) 540 arc seconds when reduced performance is accepted.
With these differences, it can be concluded from table 4.6 that the errors induced by the projection distortions are insignificant, even for worst case scenarios. It can also be noted that instead of taking the previous star tracker estimate as coarse estimate, one could use an estimated attitude from a state estimator, which might be updated at a higher rate using e.g. a gyroscope. This would further decrease the difference between the attitude at which the database image is taken and the true attitude.
The results of this table can be used to assess which amount of attitude difference is allowed before a new database image is selected. To increase the speed, the same database image should be used as long as possible, in order to eliminate the computationally intensive coordinate conversion. A threshold value could be determined so that a new database image is created when the attitude difference exceeds e.g. 500 arc seconds, in order to keep the estimation error within 1% of the optimal estimation error. As can be seen from the real life example given above, this amount of allowed attitude difference should allow to use the database image for several exposures, even when a fast maneuver is executed. In the case of the fine pointing mode, the attitude difference remains very small (in the order of a few arc seconds), so that the same database image can be used for a very long time and induced errors are definitely insignificant.

Accuracy results

The accuracy of AIM, QUEST and q-Davenport was tested during simulations of 10,000 star images distributed randomly over the sky. The values that are given for each algorithm and each simulation scenario are the root mean square errors around the three axes. For QUEST, no Newton-Raphson iterations were used, since these are not necessary to improve accuracy according to [112]. For AIM, the difference between the attitude at which the image was taken and the attitude at which the database stars were selected, was chosen to be 100 arc seconds around each axis. In other words, the coarse estimate of the attitude, q_{dat} was rotated 100 arc seconds around each axis from the known true attitude. In section 4.3.2, it was calculated that this is the largest difference (worst case scenario) at which the example star tracker is still required to offer full performance. In realistic scenarios, this difference will be a lot smaller.
The experiments of which the results are depicted in table 4.7, show that the accuracy of AIM is almost exactly the same as the accuracy of QUEST and q-Davenport. The deviation is in the orders of one hundredths of an arc second. While QUEST is expected to perform equally good as q-Davenport, there is a slight reduction in accuracy. This is because the systematic error induced by the camera distortions lowers the accuracy of QUEST, which is less robust

Table 4.7: The rms attitude angle errors of AIM, QUEST and q-Davenport (in arc seconds) in various scenarios.

scenario	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 9$	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.1$ $n_s = 9$	$\gamma = 8$ $n_{pix} = 2048$ $E_{cent} = 0.5$ $n_s = 9$
AIM rms error ψ	4.94	1.00	2.57
AIM rms error θ	4.95	0.95	2.48
AIM rms error ϕ	87.99	18.86	44.96
QUEST rms error ψ	4.94	0.99	2.57
QUEST rms error θ	4.95	0.97	2.49
QUEST rms errorϕ	88.03	19.11	44.95
q-Daven. rms error ψ	4.94	0.99	2.57
q-Daven. rms error θ	4.95	0.97	2.48
q-Daven. rms error ϕ	87.98	18.87	44.94
scenario	$\gamma = 8$ $n_{pix} = 4096$ $E_{cent} = 0.5$ $n_s = 9$	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 15$	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.1$ $n_s = 15$
AIM rms error ψ	1.20	3.69	0.77
AIM rms error θ	1.24	3.74	0.74
AIM rms error ϕ	23.67	66.66	13.44
QUEST rms error ψ	1.20	3.68	0.77
QUEST rms error θ	1.26	3.76	0.77
QUEST rms errorϕ	23.62	67.34	13.42
q-Daven. rms error ψ	1.20	3.68	0.77
q-Daven. rms error θ	1.26	3.75	0.77
q-Daven. rms error ϕ	23.62	66.67	13.42

to such a systematic error. The robustness of QUEST will be discussed more in depth in section 4.3.3. If one Newton-Raphson iteration is performed, the accuracy of QUEST is equal to that of q-Davenport, at the cost of an increase in computational time. From these results, we can conclude that AIM is as accurate as the state of the art algorithms.

Accuracy improvements

One way to benefit from the large increase in computational efficiency of AIM, is to improve the accuracy of the attitude estimation. This can be done on the one hand by using the increased efficiency to increase the speed of the attitude estimation and getting the attitude estimation at a higher frequency.

This improves the performance of the entire attitude determination and control system, but this effect is hard to quantify in tests. On the other hand, since the computational load is decreased significantly, there is room to implement changes in the algorithms, such as an improved centroiding algorithm or using more stars in the tracking algorithm.
An error representation which is interesting to give an order of magnitude idea of the effect of these accuracy improvements is the Noise Equivalent Angle (NEA). This is the star tracker's ability to reproduce the same attitude provided the same optical stimulation. This error is independent of software, algorithmic errors and calibration [75]. The NEA is approximately:

$$NEA_{cross} = \frac{\gamma.E_{cent}}{n_{pix}.\sqrt{n_s}}, \tag{4.25}$$

where NEA_{cross} is the cross boresight NEA, E_{cent} is the average centroiding accuracy, i.e. the fraction of a pixel to which the centroid of a star can be determined, typically ranging from 0.05 to 0.5, and n_{pix} is the number of pixels in the image.
Equation 4.25 presents us with a convenient calculation to estimate the accuracy of the attitude estimation, given the number of pixels and FOV and the camera, the number of stars used and the centroiding accuracy.
From equation 4.25, it can be deduced that improved centroiding is linearly proportional to the accuracy. The accuracy also scales inversely proportional with the root of the number of stars used in the tracking algorithm. Using equation 4.25, it is estimated that using an extra star yields a decrease in error which can be calculated as:

$$e_d = \frac{\frac{1}{\sqrt{n_s+1}}}{\sqrt{\frac{1}{n_s}}} = \sqrt{\frac{n_s}{n_s+1}}, \tag{4.26}$$

This decrease in error is a function of the number of stars used in the tracking algorithm. This accuracy improvement was also implemented in MATLAB and verified in simulations. During these simulations, AIM determined the attitude using one or two extra stars. The results are given in table 4.8. Simulation results to validate the accuracy improvement caused by better centroiding can be found in table 4.7, where the results of simulations performed with different values for E_{cent} are shown.
The obtained error reduction corresponds well with the predicted error reduction of equation 4.26. The decrease in error of AIM compared to QUEST or q-Davenport is in the order of a few arc seconds now (around the roll axis), which is a significant improvement.

Table 4.8: The rms attitude angle errors of AIM, QUEST and q-Davenport (in arc seconds) in various scenarios where AIM uses more stars. The number of stars used by AIM is given between brackets.

γ	8	8	8	8
n_{pix}	1024	1024	1024	1024
E_{cent}	0.5	0.5	0.5	0.5
n_s	9 (10)	9 (11)	15 (16)	15 (17)
AIM rms error ψ	4.68	4.39	3.66	3.65
AIM rms error θ	4.59	4.37	3.61	3.52
AIM rms error ϕ	85.82	79.77	65.15	65.03
QUEST (1 iter.)/ q-Dav. rms error ψ	4.94	4.94	3.68	3.68
QUEST (1 iter.)/ q-Dav. rms error θ	4.95	4.95	3.75	3.75
QUEST (1 iter.)/ q-Dav. rms error ϕ	87.98	87.98	66.67	66.67

4.3.3 Robustness

A distortion in the optics of the star camera, dead pixels, or errors in the centroiding algorithm could lead to a large error in the calculation of the centroid of one or more stars. This is especially so for smaller satellites, where there is less budget to buy expensive optics and there is less redundancy and room for error checks to limit the number of calculations. It is therefore important that the attitude estimation algorithm is robust to such errors. This is validated in the first section.
As opposed to the existing attitude estimation algorithms, AIM requires a coarse estimate of the attitude (q_{dat}) to be able to estimate the attitude of the spacecraft. The estimated attitude of the previous time step can be used as a coarse estimate. During large slew maneuvers, this coarse estimate might be far off, since the spacecraft is rotating rapidly. In the second section, the robustness of AIM is validated for increasingly inaccurate coarse estimates.

Robustness to outliers

The robustness of AIM is compared to that of QUEST (with 0 iterations) and q-Davenport in MATLAB simulations using 10,000 star tracker exposures distributed randomly over the sky. In the various scenarios, one of the image stars, the so-called outlier, was given a position error with a variance which was a factor P_e higher than the position errors of the other stars. The distance

between this image star and its corresponding database star will therefore be significantly higher. The results of these simulations are presented in table 4.9.

Table 4.9: The rms attitude angle errors of AIM, QUEST and q-Davenport (in arc seconds) with outliers in the image.

γ	8	8	8	8
n_{pix}	1024	1024	1024	1024
E_{cent}	0.5	0.5	0.5	0.5
n_s	9	9	9	9
P_e	0	25	50	100
AIM rms error ψ	4.94	41.3	84	165
AIM rms error θ	4.95	39.5	81	159
AIM rms error ϕ	87.99	736	1542	3198
QUEST rms error ψ	4.94	41.3	137	208
QUEST rms error θ	4.95	39.7	87	226
QUEST rms error ϕ	88.03	811	13097	19382
q-Daven. rms error ψ	4.94	41.4	84	165
q-Daven. rms error θ	4.95	39.5	81	159
q-Daven. rms error ϕ	87.98	736	1541	3197

The results in table 4.9 show that AIM is as robust to distortions as is q-Davenport, which is one of the robust methods. QUEST performs worse than AIM or q-Davenport in the presence of outliers. For the roll error, this difference in error can amount to more than a degree. This problem QUEST has with robustness is solved by performing two Newton-Raphson iteration. In simulations, QUEST then performed similar to q-Davenport and AIM, at the cost of an increase in the number of calculations.

Robustness to an inaccurate database attitude, q_{dat}

In tracking mode, the attitude estimation of the previous time step can be used as a coarse estimation for the attitude in the current attitude estimation. During large slew maneuvers, the spacecraft may have significantly rotated away from the attitude of the previous time step. Because of this, the coarse estimate of the attitude may be far off. The effect of this was verified by validating the performance of AIM when the coarse estimate was increasingly inaccurate, or in other words, when the spacecraft had increasingly rotated away from the previously estimated attitude. The results are given in the table below, where α_r is the angle in arc seconds over which the coarse estimate, q_{dat}, is rotated away from the true attitude over each of the three axes.
From table 4.10, it is clear that for increasingly inaccurate coarse estimates, q_{dat}, the accuracy of AIM decreases. However, even for an α_r of 10800 arc

Table 4.10: The rms attitude angle errors of AIM, QUEST and q-Davenport (in arc seconds) when q_{dat} is inaccurate.

scenario	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 9$ $\alpha_r = 0$	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 9$ $\alpha_r = 500$	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 9$ $\alpha_r = 1000$
AIM rms error ψ	4.94	4.97	5.12
AIM rms error θ	4.95	5.04	5.24
AIM rms error ϕ	87.99	93.2	95.6
QUEST rms error ψ	4.94	4.88	4.92
QUEST rms error θ	4.95	5.01	5.08
QUEST rms error ϕ	88.03	93.0	95.3
q-Daven. rms error ψ	4.94	4.88	4.92
q-Daven. rms error θ	4.95	5.01	5.08
q-Daven. rms error ϕ	87.98	93.0	95.3
scenario	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 9$ $\alpha_r = 3600$	$\gamma = 8$ $n_{pix} = 1024$ $E_{cent} = 0.5$ $n_s = 9$ $\alpha_r = 10800$	
AIM rms error ψ	7.37	35.1	
AIM rms error θ	6.93	21.6	
AIM rms error ϕ	110.43	316.1	
QUEST rms error ψ	4.99	6.33	
QUEST rms error θ	5.05	6.14	
QUEST rms error ϕ	103.9	147.8	
q-Daven. rms error ψ	4.99	6.33	
q-Daven. rms error θ	5.05	6.14	
q-Daven. rms error ϕ	103.9	147.8	

seconds, AIM still converges to a reasonable solution. Considering the real life example of the star tracker with the update frequency of 10 Hz, an α_r of 10800 arc seconds would result in the spacecraft rotating $30°/s$ around each of its three axes. At this point, it is likely that the star identification algorithm would fail since these algorithms generally also use a coarse estimate of the attitude to efficiently identify the stars [105]. In this case, the attitude estimation procedure would fail regardless of the used estimation algorithm.

It can be concluded that in normal operating conditions, using the previous

attitude as q_{dat} will not pose robustness issues, but might reduce the accuracy when large maneuvers are performed.

4.4 Conclusion

In this paper, a novel attitude estimation algorithm, referred to as AIM (Attitude estimation using Image Matching), was proposed. At the base of AIM lies an algorithm which optimally maps the stars of two images on top of each other. Since this optimization problem can be reduced to explicit equations from which the unknown variables can be calculated, AIM is both extremely fast and robust. When AIM is compared to the fast-iterative and the robust attitude estimation algorithms which exist now, it is clear that AIM is faster than the fastest algorithms, as robust as the most robust methods and has similar accuracy as the existing algorithms. Furthermore, AIM allows in a lot of cases to eliminate a very computationally expensive coordinate conversion in the algorithms preceding the attitude estimation algorithm. This way, the computational cost of the attitude estimation is reduced very significantly.
The proposed algorithm can allow to improve the performance of the attitude estimation by using the reduction in computational cost to acquire the attitude estimates at a higher rate, use more stars in the attitude estimation algorithm or improve the centroiding algorithm. It is also a valuable contribution to the expanding field of small satellite projects, where platforms have limited computational capability and the reduced computational complexity of AIM could allow the implementation of star trackers on these platforms.

Chapter 5

Robustness and Efficiency Improvements for Star Tracker Attitude Estimation

The contents of this chapter were published in the Journal of Guidance, Control and Dynamics: T. Delabie, J. D. Schutter, and B. Vandenbussche. Robustness and efficiency improvements for star tracker attitude estimation. Journal of Guidance, Control and Dynamics, 38(11):2108-2121, 2015.

This chapter expands on the work presented in chapter 4. The parts of chapter 4 that are relevant for the current chapter are briefly summarized to ensure that this chapter can be read independently. The summarizing parts are highlighted in an emphasized font (like this) and can be ignored by the reader who has read chapter 4.

In this chapter, a star tracker attitude estimation procedure with increased robustness and efficiency, using the AIM (Attitude Estimation using optimal Image Matching) algorithm, is presented and validated. The unique approach of the AIM algorithm allows to introduce a reliable quality check which can be efficiently calculated. Unlike existing validation methods, this quality check not only detects that some of the data is unreliable, it also determines which star measurements are unreliable. These unreliable measurements can be removed from the data set and a new attitude quaternion can be calculated without having to repeat the entire AIM algorithm. This greatly improves the robustness of the attitude estimation, while limiting the computational expense. Furthermore, the structure of AIM allows to reuse previously calculated data when the change in attitude between subsequent measurements is small. This way, the efficiency

of the entire attitude estimation cycle can be increased significantly. These enhancements are validated with simulated star tracker data. The results show that the improvements significantly improve the robustness and lower the computational cost of the star tracker attitude estimation. As a consequence, the overall performance of the attitude determination and control system greatly increases.

5.1 Introduction

Accurate knowledge of the orientation of the spacecraft in space is crucial in a wide variety of spacecraft missions. Examples include space telescopes observing an astronomical target or communication satellites that need to accurately point an antenna to a ground station. Of the many types of sensors that are used to determine this orientation, also referred to as the attitude of the spacecraft, the star tracker is the most accurate [115]. This sensor takes an image of the surrounding star field with an on-board camera, and compares it to a database of known star positions. This way, the star tracker can typically determine the attitude of the spacecraft in the range of a few arcseconds or even sub-arcsecond [95].

An autonomous star tracker operates in two different modes [75]. The first of these is the initial attitude acquisition, in which the star tracker determines the attitude without a priori knowledge [120, 26]. Once an initial attitude has been acquired, the star tracker has a priori knowledge and it can switch to the tracking mode. In this second mode, the a priori knowledge allows us to reduce the database search, resulting in a faster and more accurate attitude estimation. A variety of algorithms that estimate the spacecraft attitude in this tracking mode, has been proposed. The performance of these algorithms is generally judged based on their computational complexity, accuracy and robustness to inaccurate data *[79, 17]. The existing algorithms estimate the spacecraft attitude from vector measurements by minimizing a cost function, called the Wahba cost function [79]. These algorithms determine the optimal quaternion [87]. They therefore have similar accuracy and differ only in computational complexity and robustness. Based on the method to solve Wahba's problem, the existing algorithms can be divided into two main groups. Davenport's q method [77] and the SVD (Singular Value Decomposition) method [71] are based on robust algorithms to calculate the symmetric eigenvalue problem and the SVD [60, 51]. These algorithms are therefore robust to distortions but have a high computational cost. The second group solves the characteristic equation for the maximum eigenvalue and this way presents a fast iterative solution to construct the optimal attitude quaternion. The first of these methods was QUEST (QUaternion ESTimator) [111, 114], followed by FOAM (Fast Optimal Attitude Matrix) [78], ESOQ (EStimator of the Optimal Quaternion) [86], and ESOQ2 [87]. These methods*

have a comparable computational cost which is lower than that of the first group [17], but are not based on the same robust algorithms.

An important issue facing these algorithms is the possible presence of inaccurate input data, referred to as 'outliers'. Because of pixel defects, cosmic events or errors in preceding algorithms, the estimated centroid of one or more of the camera image stars could be significantly far off, leading to these outliers. This inaccurate input data greatly reduces the accuracy of the attitude estimate [24]. In order to tackle this problem, a quality check called the TASTE test was presented by the author of QUEST [113]. This test reliably signals whether or not an outlier is present in the data. A drawback of this test is that it is impractical to identify which of the stars are the outliers. It is therefore not straightforward to remove the outliers and recalculate the attitude to improve robustness.

A second important issue of the existing algorithms is the fact that they use vector measurements to determine the spacecraft attitude. In a star tracker, the observations are 2D star centroids on the focal plane. The conversion of these 2D coordinates to unit vectors is computationally complex. Because of optical and electronic distortions, temperature, magnetic and star intensity effects, an empirical model based on laboratory calibrations is often used to convert the coordinates, leading to a significantly larger computational cost [42].

Recently, a new algorithm, referred to as AIM (Attitude estimation using optimal Image Matching), was proposed by the first author of this work. This algorithm uses a different approach to estimate the spacecraft attitude. It is based on focal plane coordinates instead of unit vectors. In [24], it was shown that the accuracy of AIM is comparable to that of the state of the art algorithms, it is as robust as the robust algorithms (such as Davenport's q method) and it is less computationally complex than the fast methods (such as QUEST). The novel approach of AIM leads to new possibilities, which can greatly improve both the robustness to outliers in the data and the speed of the attitude estimation procedure.

To increase the robustness, a quality check, which can be efficiently calculated, is introduced for the AIM algorithm. This quality check not only detects whether or not there are outliers in the data, it also allows us to identify the outlier. It is shown that the outlier can be removed from the data and the attitude can be recalculated without having to repeat the entire AIM algorithm. Simulations show that this greatly improves the robustness of the attitude estimation and allows us to continue using the star tracker attitude estimates even when there are outliers.

To increase the computational speed, previously calculated data can be reused in the AIM algorithm, eliminating complex calculations. When the attitude changes between subsequent time steps are sufficiently small, the computationally complex coordinate conversion can be eliminated from the

procedure. This was already briefly mentioned in the previous chapter. In this chapter we validate the effect of this improvement.
The improvements presented in this chapter greatly increase the robustness and decrease the computational cost of the star tracker attitude estimation procedure using AIM. This way, the performance of the entire attitude determination and control system is greatly improved. The lower computational cost could allow us to carry a more demanding payload or lower the cost of the spacecraft. These improvements help facilitate the use of star trackers in the growing market of small satellites.
In section 5.2, a summary of the state of the art algorithms and AIM is given, with an emphasis on the differences in approach of these algorithms. Section 5.3 discusses the enhancements: the AIM quality check to increase robustness and the reuse of previously calculated information to increase the speed of the attitude estimation procedure. The effect of these improvements is validated in tests on simulated data. The test procedure is discussed in section 5.4 and results are given in section 5.5. The performance of the attitude estimation procedure using AIM is compared to that of a state of the art fast method (QUEST) and a robust method (the SVD method).

5.2 AIM and the State of the Art Algorithms

The improvements that are discussed in this paper can be introduced because AIM has a different approach to solve the attitude estimation problem than the state of the art algorithms. In order to discuss these differences in approach, it is necessary to look at the entire star tracker attitude estimation procedure, of which the attitude estimation algorithm is the final step in a sequence of algorithms. After this, the procedure of AIM and the state of the art algorithms is discussed further in detail.

5.2.1 Attitude Estimation Procedure

To discuss the differences between the procedure using AIM and the procedure using the state of the art algorithms, an overview of the different steps in the star tracker attitude estimation procedure is given. This sequence is depicted in figure 5.1.

The first algorithm is the centroiding algorithm *[98, 103], which locates the centroids of stars in the star camera image as accurately as possible. Once the positions of the stars are known, the camera stars are paired with their corresponding database stars in the* pairing *step [105]. Up to this point, the procedure of AIM and the state of the art algorithms are similar. In the following step, the* coordinate conversion *step, the approaches differ. This is shown in figure 5.2.*

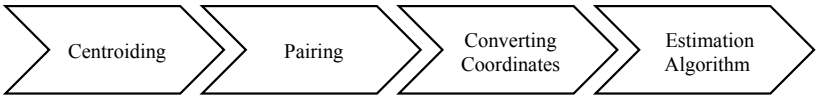


Figure 5.1: The sequence of Algorithms used to estimate the attitude with a star tracker.

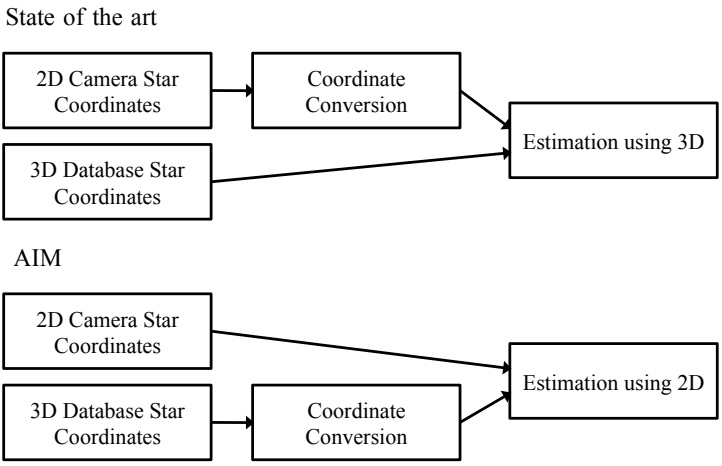


Figure 5.2: A schematic overview of the different approaches to estimate the attitude of both AIM and the state of the art algorithms.

The state of the art algorithms use unit vectors to determine the spacecraft attitude. The 2D focal plane coordinates which were measured by the star tracker camera therefore need to be converted to 3D unit vectors. AIM uses 2D focal plane coordinates to determine the attitude of the spacecraft. The 3D database star coordinates therefore need to be converted to 2D focal plane coordinates. In other words, an image of the database is generated. The coordinate conversion step and the actual attitude estimation algorithm are explained in further detail for the state of the art algorithms and for AIM in the following sections.

5.2.2 State of the Art Algorithms

In this section, the coordinate conversion and the attitude estimation procedure using the state of the art algorithms is discussed.

Coordinate Conversion

The conversion of 2D focal plane coordinates to 3D unit vector coordinates can be done using a simple pinhole model as described in [42]:

$$\begin{cases} i = \frac{\frac{x-x_0}{F}}{\sqrt{1+\left(\frac{x-x_0}{F}\right)^2+\left(\frac{y-y_0}{F}\right)^2}} \\ j = \frac{\frac{y-y_0}{F}}{\sqrt{1+\left(\frac{x-x_0}{F}\right)^2+\left(\frac{y-y_0}{F}\right)^2}} \\ k = \frac{1}{\sqrt{1+\left(\frac{x-x_0}{F}\right)^2+\left(\frac{y-y_0}{F}\right)^2}} \end{cases} \tag{5.1}$$

In these equations, (x, y) is the coordinate of the star in the image, (x_0, y_0) is the intersection of the focal plane and the optical axis, (i, j, k) is the unit vector of the observed star, and F is the focal length of the optical system.

To account for distortions however, the image plane coordinates are first corrected for distortions using a fifth order polynomial [42]:

$$x_c = -k_0 + k_1 x + k_2 y + k_3 x(x^2+y^2) + k_4 x(x^2+y^2)^2 - k_5 x^2 - k_6 xy - k_7 y^2 \tag{5.2}$$

$$y_c = -h_0 + h_1 y + h_2 x + h_3 y(y^2+x^2) + h_4 y(y^2+x^2)^2 - h_5 y^2 - h_6 yx - h_7 x^2 \tag{5.3}$$

Where (x_c, y_c) are the coordinates after they have been corrected for camera distortions, and the parameters (k_i, h_i) depend on the camera. The coordinates (x_c, y_c) are then converted using equations 5.1.

Attitude Estimation

After the camera star coordinates are converted, the attitude estimation algorithm is called. In the state of the art methods, Wahba's problem is solved to determine the spacecraft attitude:

$$L(A) = \frac{1}{2}\sum_{i=1}^{n_s} w_i |\mathbf{b}_i - A\mathbf{r}_i|^2, \tag{5.4}$$

where n_s is the number of star pairs used, $\mathbf{b}_i$ are the unit vectors observed in the spacecraft body frame, $\mathbf{r}_i$ are the corresponding unit vectors in a reference frame and w_i are non-negative weights that can be used to increase or decrease the importance of certain star pairs in the loss function.

A good overview of existing algorithms that solve Wahba's problem can be found in [79]. During the simulations in this paper, QUEST was used as an example of one of the fast algorithms, while the SVD method was used as an example of the more robust methods.

5.2.3 AIM

In this section, the coordinate conversion and the attitude estimation procedure using AIM is discussed.

Coordinate Conversion

In the AIM coordinate conversion step, the 3D database coordinates are converted to 2D focal plane coordinates. To be able to do this, a coarse estimate of the star tracker attitude is needed. This coarse attitude estimate will be referred to as quaternion q_{dat}. In tracking mode, the previous attitude estimate can be used as this estimate. The database unit vectors are then rotated over the inverse of q_{dat} so that they are centered around the k-axis, as shown in equation 5.5. Since q_{dat} is a unit quaternion, the inverse is equal to its conjugate.

$$\begin{pmatrix} \hat{i}_r \\ \hat{j}_r \\ \hat{k}_r \end{pmatrix} = \bar{q}_{dat} \begin{pmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{pmatrix} q_{dat} \tag{5.5}$$

In this equation, $(\hat{i}, \hat{j}, \hat{k})$ are the unit vector coordinates of a star as found in the database, $(\hat{i}_r, \hat{j}_r, \hat{k}_r)$ are those unit vector coordinates after rotation by the inverse of q_{dat}, and a bar over a quaternion indicates that the conjugate of the quaternion is taken.

After this, the conversion from unit vector coordinates to database focal plane coordinates $(\hat{x}, \hat{y})$ is performed. Using the pinhole model, this conversion is performed with the following equations:

$$\begin{cases} \hat{x} = x_0 + F\hat{i}_r/\hat{k}_r \\ \hat{y} = y_0 + F\hat{j}_r/\hat{k}_r \end{cases} \tag{5.6}$$

The resulting coordinates $(\hat{x}, \hat{y})$ are the focal plane coordinates an ideal star tracker would observe if its attitude was q_{dat}. Because a real star tracker has camera distortions, these coordinates are transformed to account for these distortions. Using the inverse transformation of equations 5.2 - 5.3, coordinates $(\hat{x}, \hat{y})$ are transformed to the values as they would be measured with the distortions in the camera. The polynomials to calculate this transformation are given in equations 5.7 and 5.8.

$$\begin{aligned} \hat{x}_d = &-k_{0in} + k_{1in}\hat{x} + k_{2in}\hat{y} + k_{3in}\hat{x}(\hat{x}^2 + \hat{y}^2) + \\ &k_{4in}\hat{x}(\hat{x}^2 + \hat{y}^2)^2 - k_{5in}\hat{x}^2 - k_{6in}\hat{x}\hat{y} - k_{7in}\hat{y}^2 \end{aligned} \tag{5.7}$$

$$\begin{aligned} \hat{y}_d = &-h_{0in} + h_{1in}\hat{y} + h_{2in}\hat{x} + h_{3in}\hat{y}(\hat{y}^2 + \hat{x}^2) + \\ &h_{4in}\hat{y}(\hat{y}^2 + \hat{x}^2)^2 - h_{5in}\hat{y}^2 - h_{6in}\hat{y}\hat{x} - h_{7in}\hat{x}^2 \end{aligned} \tag{5.8}$$

In these equations, $(\hat{x}_d, \hat{y}_d)$ are the database star coordinates after the camera distortions have been added, and (k_{iin}, h_{iin}) are the parameters which are specific for the camera.

Attitude Estimation

After the database star coordinates are converted, AIM is called. AIM estimates the attitude by calculating the transformation values ϕ, t_x, and t_y, which minimize the following cost function [24]:

$$\Gamma(\phi, t_x, t_y) = \sum_{i=1}^{n_s} w_i \left((x_i - \hat{x}_i cos(\phi) + \hat{y}_i sin(\phi) - t_x)^2 + (y_i - \hat{x}_i sin(\phi) - \hat{y}_i cos(\phi) - t_y)^2 \right) \tag{5.9}$$

In this function, w_i is the weight given to star pair i, (x_i, y_i) are the coordinates of observed star i in the focal plane, $(\hat{x}_i, \hat{y}_i)$ are the coordinates of the corresponding database star i in the focal plane, ϕ is the angle over which the database stars are rotated with respect to the origin of the frame in which the database coordinates are described, and t_x and t_y are the distances over which the database stars are translated in x- and y-direction respectively.

Minimizing the cost function of AIM (equation 5.9), leads to the following expressions to calculate the transformation values ϕ, t_x, and t_y [24]:

$$\phi = atan2(sx.s\hat{y} - sy.s\hat{x} - sx\hat{y} + sy\hat{x}, -sx.s\hat{x} - sy.s\hat{y} + sx\hat{x} + sy\hat{y}) \tag{5.10}$$

$$t_x = sx.cos(\phi) + sy.sin(\phi) - s\hat{x} \tag{5.11}$$

$$t_y = -sx.sin(\phi) + sy.cos(\phi) - s\hat{y} \tag{5.12}$$

The variables in these three equations are calculated as follows:

$$
\begin{aligned}
sx &= \sum_{n=1}^{n_s} w_i.x_i & sx\hat{x} &= \sum_{n=1}^{n_s} w_i.x_i.\hat{x}_i \\
sy &= \sum_{n=1}^{n_s} w_i.y_i & sx\hat{y} &= \sum_{n=1}^{n_s} w_i.x_i.\hat{y}_i \\
s\hat{x} &= \sum_{n=1}^{n_s} w_i.\hat{x}_i & sy\hat{x} &= \sum_{n=1}^{n_s} w_i.y_i.\hat{x}_i \\
s\hat{y} &= \sum_{n=1}^{n_s} w_i.\hat{y}_i & sy\hat{y} &= \sum_{n=1}^{n_s} w_i.y_i.\hat{y}_i
\end{aligned}
\tag{5.13}
$$

Based on these transformation values, the attitude of the spacecraft can be estimated. The rest of the procedure to do this is of no relevance in this chapter and can be found in the previous chapter.

5.3 Improved Attitude Estimation using AIM

This section discusses the adaptations that can be made to the attitude estimation procedure using AIM, in order to increase the robustness and the speed of the attitude estimation. In the first part of this section, the robustness of the attitude estimation to the so-called 'outliers' is discussed. A method is presented which not only detects whether inaccurate star data is present in the data set, but also determines which of the stars are the outliers. This allows us to remove the inaccurate data and recalculate the attitude.
In the second part of this section, a method is proposed to increase the computational efficiency of the attitude estimation using AIM. This is done by reusing previously calculated data when the change in attitude between subsequent attitude determination steps is small. When the attitude changes are small, this method allows us to eliminate a computationally expensive coordinate conversion, hereby greatly increasing the efficiency of the entire attitude estimation procedure.

5.3.1 Robustness

The centroiding algorithm [124, 131], which precedes the tracking algorithm, estimates the centroid of each star in the camera image. This estimate is subject to noise [129]. In some cases, because of dead pixels, hot pixels [41], an error in the centroiding algorithm or other effects, the centroiding error of one or more stars may be significantly higher than that of the other stars. Such a

star with a higher positional error will be referred to as an 'outlier'. These outliers significantly lower the accuracy of the attitude determination. A quality check to detect these outliers has been developed [113] for the state of the art algorithms, but the downside is that it only flags whether or not an outlier is present. In the following section, a method is developed which not only signals that an outlier is present, it also indicates which star is the outlier. Furthermore, an efficient procedure is presented which efficiently recalculates the attitude when an outlier has been found. This method is presented for both the state of the art algorithms as for the AIM algorithm.
First, the existing method to detect outliers is presented, the TASTE test [113]. After this, the procedure to detect and remove outliers is shown for both the state of the art algorithms as for the AIM method.

Robustness improvements: State of the Art

In this section, we will discuss how inaccurate star data can be detected and removed in the attitude estimation procedure using one of the state of the art algorithms.

Detecting outliers: TASTE Before an efficient and automatic test was developed to detect outliers, data validation was carried out by an analyst who manually fitted a curve to the data to remove outliers [113]. After the development of the QUEST attitude estimation algorithm, a data checking method was presented which signals whether or not there are outliers in the image. This method is called the TASTE test and can be calculated very efficiently because the values needed in this test are central to the attitude computation [112]. The TASTE test is calculated as depicted in equation 5.14:

$$TASTE = 2(\lambda_0 - \lambda_{max}). \tag{5.14}$$

In this equation, λ_0 is the sum of the weights given to the star pairs in the cost function and is chosen to be equal to 1. λ_{max} is the maximum eigenvalue which is calculated in the QUEST attitude calculation.
When one or more stars in the camera image have a significantly larger positional error, the TASTE value becomes very large [113], allowing to detect the presence of an outlier. According to the creator of TASTE, more than the computational efficiency of the QUEST algorithm, it was this efficient TASTE test which was of importance from a mission perspective [112].

Detecting outliers: Distance method In the state of the art methods, which solve Wahba's problem, the distance between the unit vectors in the camera frame and the transformed unit vectors in the database frame is minimized by finding the transformation matrix A. Once this matrix A, or a corresponding quaternion, has been determined by QUEST, one can use this quaternion to

rotate the unit vectors in the database frame. For a perfect attitude estimate, this would result in transformed unit vectors which are equal to the unit vectors in the camera frame. In general, because of noise, the unit vectors will all deviate to a certain extent. When an outlier is present in the data, the transformed unit vector of that star will deviate considerably more and the distance between that transformed database vector and its corresponding camera vector will be significantly larger. This procedure, which can be used to detect which stars are the outliers, is discussed next.
The database unit vectors are first rotated over the estimated attitude quaternion, q_e, which has just been determined.

$$\mathbf{r}_{it} = q_e \mathbf{r}_i \bar{q}_e \tag{5.15}$$

The angle between the camera unit vectors and the transformed database unit vectors is selected as the distance. This angle can be calculated easily as follows:

$$\theta_d = acos(\mathbf{b}_i \cdot \mathbf{r}_{it}), \tag{5.16}$$

where θ_d is the angle between the unit vectors, $\mathbf{b}_i$ is the measured unit vector in the camera frame and $\mathbf{r}_{it}$ is the transformed database vector. In order to speed up this procedure, the computationally complex acos function can be discarded. In order to get a distance measure which is zero in case of a perfect star pair, the distance measure is then calculated as:

$$d = 1 - \mathbf{b}_i \cdot \mathbf{r}_{it}. \tag{5.17}$$

This distance will be significantly larger for an outlier. In case this distance is larger than a certain threshold distance, the algorithm signals the star pair as an outlier. This threshold distance could be determined by selecting a value for the maximum angle allowed between the camera vectors and corresponding database vectors:

$$d_{max} = 1 - cos(\theta_{dM}). \tag{5.18}$$

This procedure to detect outliers is more computationally complex than the TASTE test. A possibility to speed up the entire attitude estimation procedure, while maintaining high robustness, is to calculate the TASTE test first and to only use the Distance method when the TASTE value is above a certain threshold value. This way, the algorithm only looks for outlier stars when the TASTE test indicates that there probably are outliers in the data.

Removing outliers When an outlier has been found, it can be removed from the data set and the attitude quaternion is calculated again. In the state of the art methods, such as QUEST and the SVD method, one of the first steps is to calculate a matrix B from the star vectors:

$$B = \sum_{1}^{n_s} w_i \mathbf{b}_i \mathbf{r}_i^T \tag{5.19}$$

To remove the inaccurate data of one outlier star, the following procedure is used:

$$B_r = B - w_r \mathbf{b}_r \mathbf{r}_r^T. \tag{5.20}$$

Here, B_r is the new B matrix that is obtained after removing the outlier, w_r is the weight assigned to the star pair with the outlier, $\mathbf{b}_r$ is the unit vector of removed star r, observed in the spacecraft body frame and $\mathbf{r}_r$ is the unit vector of removed star r, observed in a reference frame. Because the sum of the weights after removal of this star is no longer equal to 1, this matrix has to be rescaled:

$$B_r = B_r \frac{1}{1 - w_r} \tag{5.21}$$

The spacecraft attitude can then be recalculated using this new matrix B. After this, the result is checked again for outliers. This procedure is repeated until no more outliers are found. If a maximum number of outliers is detected, the attitude estimate is flagged as unreliable.

Robustness Improvements: AIM

In this section, we will discuss how inaccurate star data can be detected and removed in the attitude estimation procedure using AIM.

Detecting outliers: Distance method In the attitude estimation procedure using AIM, outliers are detected by calculating the euclidean distance squared between each pair of camera image star and transformed database star. When there is an outlier in the data, the distance between that star and its corresponding transformed database star will be significantly larger. This procedure therefore allows us not only to detect outliers, but also to detect which of the stars are the outliers. This allows us to remove the outlier from the input data and recalculate the attitude quaternion. This way, a valid attitude quaternion can be obtained, regardless of the presence of an outlier.
After the AIM attitude calculation has been executed, the database star focal plane coordinates are first transformed using the calculated transformation values (equations 5.10-5.12). Since the rotation angle ϕ is small [24], a first order approximation is used for the goniometric functions:

$$\hat{x}_{it} = \hat{x}_i - \hat{y}_i\phi + t_x \tag{5.22}$$

$$\hat{y}_{it} = \hat{x}_i\phi + \hat{y}_i + t_y; \tag{5.23}$$

In this equation, $(\hat{x}_i, \hat{y}_i)$ are the coordinates of database i, ϕ, t_x and t_y are the transformation values calculated in equations (5.10-5.12) and $(\hat{x}_{it}, \hat{y}_{it})$ are the transformed database coordinates of star i.
The euclidean distance squared between the image star coordinates and the transformed database coordinates is then determined:

$$d_i = (x_i - \hat{x}_{it})^2 + (y_i - \hat{y}_{it})^2, \tag{5.24}$$

where (x_i, y_i) are the coordinates of the camera image star i. This distance squared is in fact the value of the cost function (equation 5.9) for star pair i (with weight w_i set to 1).
The distance squared will be significantly higher for an outlier. When this distance is higher than a predetermined value, the star is considered to be an outlier. This predetermined value could be based on the expected distance between the detected star and the transformed database star.
To calculate the expected distance, it is first assumed that the noise on the centroid position of the star is Gaussian white noise. This noise depends on the used detector, optics and centroiding algorithm, and has a standard deviation of E_{cent}, expressed in fraction of a pixel. When the star tracker has n_{pix} pixels, star coordinates described between $\frac{-l}{2}$ and $\frac{l}{2}$ in the y-direction and between $\frac{-b}{2}$ and $\frac{b}{2}$ in the x-direction, we expect the noise on the centroid position to have the following standard deviation:

$$\sigma_{pos} = \sqrt{\left(\frac{E_{cent}l}{n_{pix}}\right)^2 + \left(\frac{E_{cent}b}{n_{pix}}\right)^2}. \tag{5.25}$$

In the simulations which are discussed in the following section, a star was considered to be an outlier if the distance between the image star and corresponding transformed database star exceeded the following value:

$$\sqrt{d} > 3 * \sigma_{pos} = \sqrt{\left(\frac{3E_{cent}l}{n_{pix}}\right)^2 + \left(\frac{3E_{cent}b}{n_{pix}}\right)^2}. \tag{5.26}$$

Removing outliers When an outlier has been found, it is removed from the data set and the attitude quaternion is calculated again. Recalculating the quaternion can be done very efficiently without having to repeat the entire attitude estimation algorithm. To remove the data of one outlier star, the following procedure is used:

$$\begin{aligned} sx &= sx - w_r.x_r & sx\hat{x} &= sx\hat{x} - w_r.x_r.\hat{x}_r \\ sy &= sy - w_r.y_r & sx\hat{y} &= sx\hat{y} - w_r.x_r.\hat{y}_r \\ s\hat{x} &= s\hat{x} - w_r.\hat{x}_r & sy\hat{x} &= sy\hat{x} - w_r.y_r.\hat{x}_r \\ s\hat{y} &= s\hat{y} - w_r\hat{y}_r & sy\hat{y} &= sy\hat{y} - w_r.y_r.\hat{y}_r. \end{aligned} \tag{5.27}$$

In these equations, w_r is the weight assigned to the star pair with the outlier, (x_r, y_r) are the camera image coordinates of the outlier star and $(\hat{x}_r, \hat{y}_r)$ are the coordinates of its corresponding database star.
The transformation values and the attitude quaternion are then recalculated using these new values. The result is then checked again for outliers. This procedure is repeated until no more outliers are found. If a maximum number of outliers is detected, the attitude estimate is flagged as unreliable.

5.3.2 Efficiency

An increased efficiency of the algorithms leads to a lower computational cost of the attitude estimation process. This allows the control system to achieve the attitude information at a higher rate [87], increasing the performance of the attitude determination and control system. The lower computational cost could also allow us to track more stars or use a more accurate centroiding algorithm, both of which improve the accuracy of the system. Especially in smaller satellite missions, a less computationally expensive algorithm allows us to have a more demanding payload or reduce the cost of the satellite.
An approach to increase efficiency by reusing previously calculated data is discussed next. It was already briefly introduced in [24]. Here its effect will be investigated further.

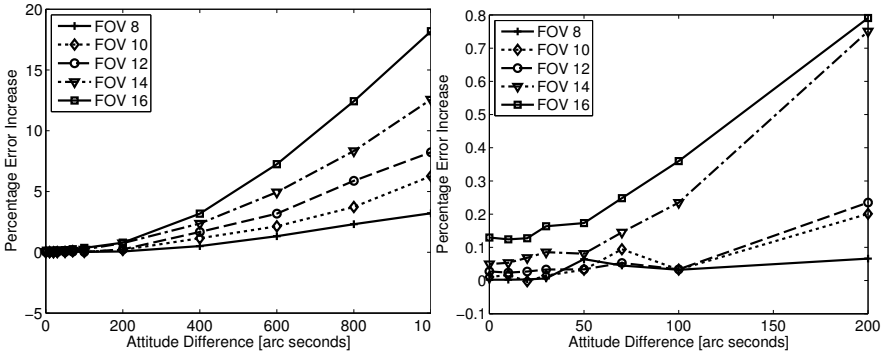
Reusing the database image

Since AIM uses the focal plane coordinates to determine the attitude, the database unit vector coordinates need to be converted to focal plane coordinates. This is different from the existing methods, where the measured camera image coordinates are converted to unit vectors. This distinction allows us to significantly increase the efficiency of the AIM attitude estimation process. Since the database coordinates do not change in each time step (as opposed to the measured camera image coordinates), the same converted database coordinates can be reused when the same stars are in view. This allows us to eliminate the coordinate conversion from 3D database unit vectors to 2D focal plane coordinates, when the image is reused. This coordinate conversion is generally computationally expensive since an empirical model based on laboratory calibrations is often used to account for distortions [42].

Reuse frequency

In order to maximize the efficiency, the same database image should be used as long as possible. A problem which arises here is that some distortion is always introduced when a 3D image is projected onto a plane [40]. This means that the position of stars relative to each other is slightly different when the star image is taken with a different camera center. The projection distortions decrease the attitude estimation accuracy when the difference between the attitude at which the camera image was taken, q_t, and the attitude at which the database stars were taken, q_{dat}, increases. In other words, the accuracy decreases when the coarse estimate of the current attitude, q_{dat}, is further off from the true attitude q_t. This can be seen in figure 5.3, where the estimation error increases when the difference between the coarsely estimated attitude q_{dat} and the true attitude q_t increases.
The attitude determination error was determined by calculating the angle in arc seconds between the estimated quaternion - q_e - and the true quaternion - q_t - as:

$$\theta_e = 2acos(\mathbf{q}_e \cdot \mathbf{q}_t)/\pi * 180 * 3600 \tag{5.28}$$



(a) (b)

Figure 5.3: The percentual increase in attitude determination error in function of the distance between q_{dat} and q_t for different field of views.

The values on the x-axis represent the angular difference between q_{dat} and q_t around all three axis. The y-axis shows the percentual increase in attitude determination error of AIM over the determination error of QUEST. A value of zero means that AIM and QUEST have the same attitude determination error. Each measurement point seen on the graphs is the result of a simulation with 10,000 images generated randomly over the sky. In these simulations, there were

9 stars in the image, the camera had 1024 pixels in each row and column, and the centroiding accuracy was 0.2 pixel. The error increases more rapidly when the FOV of the star tracker is larger. The reason is that while QUEST uses 3D unit vectors, the AIM algorithm uses 2D star coordinates and treats the star image as a 2D image. As the field of view of the star tracker increases, the assumption that the star coordinates are in a plane is increasingly inaccurate. In figure 5.3b, we see a detail of figure 5.3a.

The shown effect will limit the reuse frequency. When the same database image is used while the spacecraft is maneuvering away from the attitude at which the database image was selected, the accuracy of the attitude estimate will start to decrease. When the difference is not too large, the effect on the accuracy is negligible, but when the distance increases, the error on the attitude estimate quickly rises. In order to give an indication of the magnitude of this difference in normal operation, we determine the difference in attitude between two consecutive exposures that can be expected for an existing star tracker. From the technical specifications of the CT-602 Star Tracker of Ball Aerospace [9], it can be seen that the star tracker offers full performance up to a tracking rate of 0.3 deg/sec and reduced performance up to 1.5 deg/sec, while having an update rate of 10 Hz. Therefore, the difference in attitude between two consecutive exposures can for this star tracker maximally be (0.3 $\frac{^\circ}{s}$ *. 3600* $\frac{arcsec}{^\circ}$ */ 10* $\frac{1}{s}$ *=) 108 arc seconds with full performance and (1.5* $\frac{^\circ}{s}$ *. 3600* $\frac{arcsec}{^\circ}$ */ 10* $\frac{1}{s}$ *=) 540 arc seconds when reduced performance is accepted. At full performance (108 arc seconds difference), the error increase is clearly very small, in the order of tenths of a percent. At reduced performance, the error increase is lower than 1 percent for the FOV of 8 degrees and about 5 percent for the largest FOV.*

The quality check that was introduced in section 5.3.1, equation 5.26, can be used to determine when a database image can be reused and when a new database image should be selected. When the quality check indicates that the accuracy of the attitude estimate has deteriorated more than a predetermined threshold, new database coordinates will be converted. By changing the threshold, one can choose to have a faster, slightly less accurate estimate, which reuses database images during more time steps, or a slower but more accurate estimate of the attitude, which more rapidly creates a new database image.

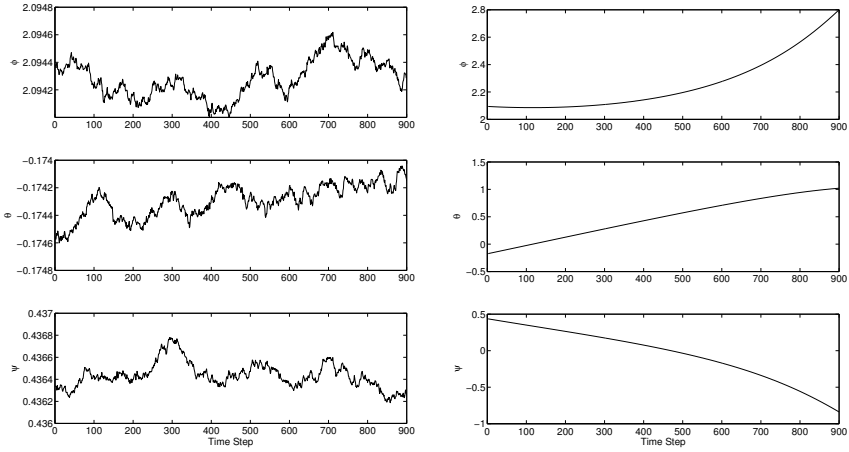
When a slew maneuver is performed, the difference between the attitude at which the image was taken, q_t and the attitude at which the database image was taken, q_{dat}, changes quite rapidly. The database image will therefore be reused less frequently. In the case of a pointing operation, the attitude of the spacecraft stays more or less fixed for a long time, allowing the same database information to be reused for a long time. Especially in this fine pointing mode, the efficiency of the spacecraft will increase significantly. The increase in efficiency was verified for different simulated maneuvers in section 5.5.2.

5.4 Test Procedure

In this section, the effect of the improvements on the attitude estimation is validated. The available input data is discussed first. After this, the attitude estimation procedure used in the tests is shown.
All tests were performed on a Dell Latitude laptop with a 2.60 GHz Intel Core i7 processor and 8 GB RAM. The C++ code was compiled by Microsoft Visual C++ 6, with the /O2 flag to maximize speed and the /ML library compiler option for static single-threaded libraries. The code was written in C++ and computation times were measured using the Windows system call QueryPerformanceCounter. The simulation environment differs from the situation on board of a satellite, where the algorithms are run on a microprocessor. However, none of the algorithms have high memory demands or benefit specifically from the high performance of the computer. The conclusions regarding computational complexity are therefore also representative for the flight environment.
The performance of AIM was compared to that of the state of the art algorithms. As an example of one of the 'fast' methods, QUEST was selected. The SVD method was selected as an example of one of the more 'robust' methods.

5.4.1 Input Data

During the tests, simulated star data was used. An advantage of using simulated star tracker data is that the true attitude is known, allowing to calculate the estimation error. Furthermore, one can verify the impact of changing various parameters, such as centroiding accuracy, slew rate, etc.
The first step of the data generation is to select the maneuver of the spacecraft. In the simulations, a difference was made between a 'pointing' maneuver and a 'slew' maneuver.
During a pointing maneuver, the spacecraft stays more or less fixed on a given attitude. Because of disturbance torques and an imperfect ADCS, the spacecraft will not maintain the attitude perfectly, but slowly move around it. In order to simulate this, the spacecraft was assumed to rotate slightly around all three axes. During the following simulations with a pointing maneuver, the magnitude of the rotational velocities around each of the three axes was chosen to be normal distributed with mean 0 and standard deviation 36 arc seconds/s. An example of a pointing maneuver is shown in figure 5.4a, where the roll (ϕ), pitch (θ) and yaw (ψ) angles are shown.
During a slew maneuver, the spacecraft rotates toward a new attitude. In the simulations, a slew around the yaw angle of the spacecraft was performed. In figure 5.4b, the roll (ϕ), pitch (θ) and yaw (ψ) angles of a slew maneuver of 1 degree per second around the yaw axis are shown.



(a) (b)

Figure 5.4: The roll (ϕ), pitch (θ), and yaw (ψ) angle for a pointing maneuver 5.4a and a slew maneuver 5.4b

.

The simulated maneuvers consist of 1000 time steps. At each of the simulated time steps, a star image was generated, using the Hipparcos Catalog [36]. Stars up to a magnitude of 5.3 were included in the image and a maximum of 9 stars was used in each image. Noise was then added to the star positions and these 'estimated' star positions were stored. The noise on the star positions was modeled as Gaussian white noise with zero mean and standard deviation on x-coordinate and y-coordinate:

$$\sigma_x = \frac{E_{cent} b}{n_{pix}} \tag{5.29}$$

$$\sigma_y = \frac{E_{cent} l}{n_{pix}} \tag{5.30}$$

On top of this noise, there is a secondary noise source because of the optics which are not perfect. This noise source was not added to the simulated data. In other words, the values for k_1, h_1, $k_{1_{inv}}$ and $h_{1_{inv}}$, were chosen to be one, while the rest of the coefficients in equations 5.2 - 5.3 and 5.7 - 5.8 were zero. The reason for not including this noise source is because the results would otherwise be dependent on a certain camera model. The attitude estimation simulations

that were performed in reference [24] have shown that the different algorithms (AIM, QUEST and SVD) are similarly affected by optical distortions. In those simulations, the coefficients were chosen to be the same of that of the Herschel Star Tracker [42] and the different attitude estimation algorithms yielded similar accuracy.
During the simulations, the following parameters for the star tracker were used:

Table 5.1: Star Tracker parameters.

FOV	16.4°
n_{pix}	1024
E_{cent}	0.2
frequency	10 Hz

5.4.2 Test setup

In this section, the attitude estimation procedures which were used in the simulations are discussed. A difference is made between the procedure when AIM reuses database images and when AIM does not reuse database images. The state of the art algorithms do not use database images but rather convert the camera image stars to unit vectors and therefore do not have these different procedures. The procedure without reuse of database images is discussed first.

Attitude Estimation Procedure

In the first step of the attitude estimation procedure, the star centroids are read from the simulated data files. In a real star tracker, the image taken by the camera would be processed and a centroiding algorithm would estimate the centroids of the stars.
The next step is to perform the star identification. In this step, the camera image stars are matched with their corresponding database stars. When there is a priori information, and the quality index of the previous attitude estimate was better than a predetermined threshold, the star tracker is in 'tracking mode'. In this mode, the a priori information is used to speed up the star identification procedure [106]. When there is no a priori information or the information cannot be trusted, the procedure is in 'lost-in-space mode'. In this case, the star identification step is performed by a lost-in-space algorithm which robustly goes through the entire database to identify the stars. For this lost-in-space algorithm, the algorithm of reference [26] is used. In the first step, the lost-in-space algorithm is always called, since there is no a priori information. From that moment on, the 'lost-in-space mode' will only be used if the attitude

has been 'lost', which is detected when a quality index surpasses a certain threshold. In general operation, a loss of attitude is unusual and the value of the threshold is quite high. In the procedure used in this paper, the quality index is checked after each attitude estimate and the 'lost-in-space mode' is entered within the same time step if the quality index is too high. The advantage of this is that the attitude estimate can be corrected within the same step if the a priori information cannot be trusted.
After the stars have been identified, the coordinates of the camera image stars are converted to unit vectors for the state of the art procedure, while the database coordinates are converted to focal plane coordinates for the AIM procedure. As the final step, the star identification algorithm is executed to determine the attitude. This procedure is executed for every time step of the measurements.

AIM Attitude Estimation Procedure with Reuse of Database Images

When AIM reuses database images to speed up the attitude estimation procedure, a separate case is added to the previous algorithm structure.
As was discussed in section 5.3.2, reusing the database information more frequently reduces the calculation time, but also leads to a deterioration of the accuracy. To determine when the information should be recalculated, the quality index presented in section 5.3.1 can be used. When the quality index is lower than a predetermined 'reuse' limit (referred to as the $ErrorLimit$), the database image can be reused. In this mode, there is no coordinate conversion. When the accuracy has deteriorated too much, the quality index of the previous time step will be higher than $ErrorLimit$ and the information needs to be recalculated. One can choose to set $ErrorLimit$ high, in order to have a faster algorithm which allows a larger deterioration in accuracy, or to choose it low which leads to a slower algorithm with higher accuracy. When $ErrorLimit$ is set equal to zero, this algorithm sequence turns into the previous algorithm sequence and no database images are reused.

5.5 Test Results

In this section, the test results are discussed. First, the robustness tests are discussed. In the first test, it is verified whether the distance measures discussed in section 5.3.1 allow us to reliably detect outliers. After this, the effect of removing outlier data on the attitude estimation error and on the computational time is verified in different scenarios for both AIM and the state of the art methods, QUEST and the SVD method. Next, the efficiency test results are discussed. The effect of reusing previously calculated database images is verified using simulated pointing and slew maneuvers.

For the simulated data, the attitude determination error in each time step was determined by calculating the angle in arc seconds between the estimated quaternion - q_e - and the true quaternion - q_t - as in equation 5.28.

5.5.1 Robustness

In this section, the proposed robustness improvements are assessed in tests with simulated data. The first test verifies the potential to detect outliers, for each of the methods discussed in section 5.3.1. In the second test, the removal of outliers is demonstrated and the effect of this removal on the attitude estimation accuracy and computation time is shown.

Detecting Outliers

In the first simulation, it is tested whether the detector methods can successfully identify outliers. In the first part of this test, the algorithms do not yet remove outliers as is presented in section 5.3.1. A pointing maneuver lasting 1000 time steps is simulated with the parameters of table 5.1 and as described in section 5.4.1. In between time step 300 and 700, one of the stars ($n_o = 1$) was given a standard deviation which was 10 times (P = 10) higher than the usual standard deviation.

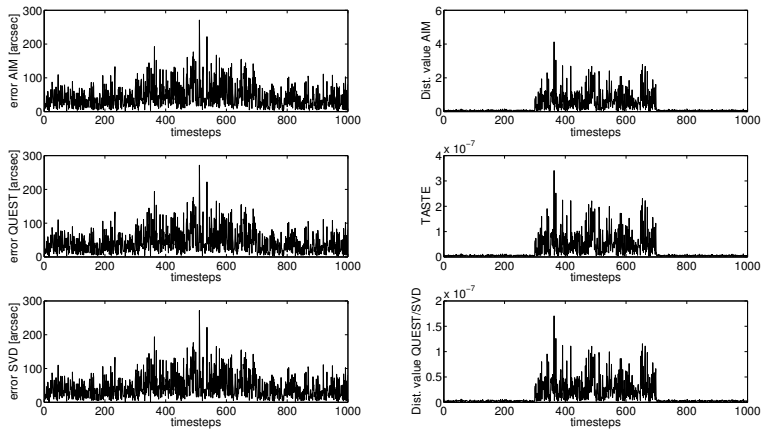


Figure 5.5: Left: The attitude estimation error clearly rises when an outlier is present. Right: The higher value of the quality checks indicate that an outlier is present.

On the left three graphs, the attitude estimation error is given for AIM, QUEST and the SVD method respectively. This error was calculated using equation

5.28. The effect of the outlier is clearly visible. Between time steps 300 and 700, the attitude estimation error is significantly higher. On the right three graphs, the results of the quality checks are given. In the top right graph, the result of the AIM Distance method is given, the middle graph shows the TASTE tests and the bottom graph shows the Distance method for the state of the art algorithms. The result for the Distance methods was obtained by taking the average of the distances between each of the star pairs. All three quality checks clearly have a significantly larger value between time steps 300 and 700, signaling that an outlier is present.
In the second part of this test, the effect of removing the outlier is studied.

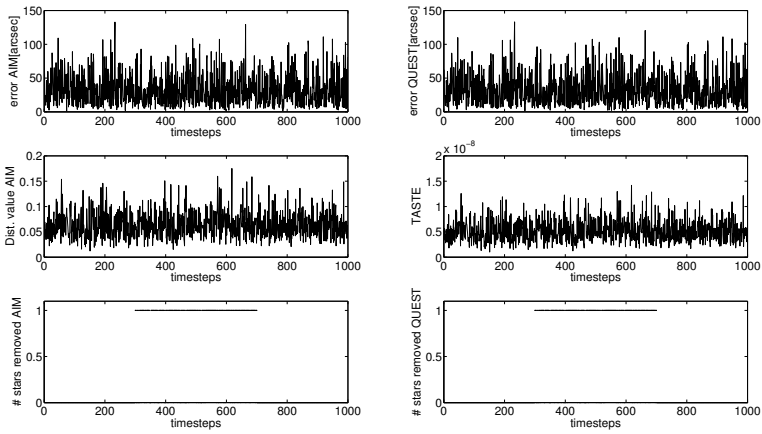


Figure 5.6: The attitude estimation error is reduced and the quality check values are normal after the outlier is removed.

The top graphs of Figure 5.6 show the attitude estimation error of AIM and QUEST (SVD has similar results as QUEST) after the outlier is removed. The error is no longer significantly higher between time steps 300 and 700. The two middle graphs show the results of the Distance method of AIM and the value of the TASTE test. Since the outlier was removed, and no other outliers are present, there are no peaks in the quality checks. The two bottom graphs show the amount of stars that were removed. Between time steps 300 and 700, during the majority of time steps, one of the stars was removed.
From these results, it follows that the Distance methods proposed in this paper are equally reliable to flag outliers in the data as the existing TASTE test. The advantage of the Distance methods is that they can also indicate which stars are the outliers. After this outlier is removed, the attitude estimation error is reduced and the quality checks no longer indicate outliers in the data.

Removing Outliers

In table 5.2 the result of removing outliers is given for different scenarios. In these scenarios, a number of stars - n_o - were turned into outliers with a standard deviation on their positional error which was a factor - P_o - higher than that of the other stars. For each scenario, 100 simulated pointing maneuvers of 1000 time steps were generated with random starting attitudes. The shown error was determined by taking the root mean square of the errors of each time step. The computational time per time step and the number of removed stars was averaged over the data.
The detection threshold to detect outlier stars for AIM was determined using equation 5.26, with the values of table 5.1. The threshold for QUEST and the SVD method was determined using equation 5.18, where θ_{dM} was chosen so that a similar number of stars would be removed as is the case for AIM.

Table 5.2: The attitude errors, computational times and number of removed stars of QUEST, SVD and AIM, with and without outlier-remover (o-r) in different scenarios.

scenario	$n_o = 0$	$n_o = 1$ $P_o = 5$	$n_o = 1$ $P_o = 10$	$n_o = 2$ $P_o = 5$
AIM rms error (arc sec.)	37.17	70.66	126.61	89.91
QUEST rms error (arc sec.)	37.18	70.63	126.68	89.86
SVD rms error (arc sec.)	37.18	70.63	126.68	89.86
AIM o-r rms error (arc sec.)	37.18	47.08	42.07	61.18
QUEST o-r rms error (arc sec.)	37.19	46.96	42.00	61.07
SVD o-r rms error (arc sec.)	37.19	46.96	42.00	61.07
AIM time (μs)	2.87	2.93	2.96	2.88
QUEST time (μs)	2.96	3.04	3.07	3.02
SVD time (μs)	8.50	8.68	8.74	8.57
AIM o-r time (μs)	3.01	3.26	3.37	3.25
QUEST o-r time (μs)	4.79	6.02	6.69	6.89
SVD o-r time (μs)	8.34	11.67	13.48	14.13
AIM % 1 star removed	0.06	55.97	86.49	3.83
AIM % 2 star removed	0	0.01	0.06	48.70
QUEST % 1 star removed	0.07	56.24	86.62	3.82
QUEST % 2 star removed	0	0.03	0.14	49.04
SVD % 1 star removed	0.07	56.23	86.61	3.82
SVD % 2 star removed	0	0.03	0.14	49.04

From table 5.2, it is clear that an outlier in the star data has a significant negative effect on the attitude estimation accuracy. The error rises significantly, even

when there is only a relatively small outlier present. This again demonstrates the importance of being able to detect and remove outliers in the data.
When the 'outlier-remover' sequence is used for the attitude estimation algorithms, the algorithms clearly perform better. The estimation algorithms are not significantly affected by the outlier star, as was the case without outlier-remover. There is however a small increase in error compared to the scenario without outliers. This is due to the fact that not all outlier stars are removed, since some are below the detection threshold. Setting the detection threshold lower would result in the removal of more outlier stars, at the cost of a longer computational time. Another reason for this small increase in error is because the algorithms estimate the attitude with fewer stars when a star is removed.
The computational times of the SVD method are significantly higher than that of AIM and QUEST, because it uses the computationally complex singular value method. The computation times are in general higher for the algorithms with outlier-remover, since they perform extra calculations to detect and remove the outliers. This increase in computational time becomes more significant when more outliers or more severe outliers are present, because more outliers are detected and removed then. An important thing to note is that the computational time of AIM does not increase as much as for QUEST or the SVD method when the 'outlier-remover' is added. The calculation time of the SVD method rises significantly when outliers are removed, because the computationally expensive singular value decomposition method is repeated. The large increase in computational time for QUEST is mainly due to the calculations that are performed to check for outliers. When QUEST is used with outlier-remover, it can no longer only rely on the TASTE test, because this test does not indicate which of the stars are the outliers. With outlier remover, QUEST needs to start using the Distance method, which is computationally more expensive. The calculation of the distances, together with the removal of outlier data and recalculation of the attitude, clearly increase the computational time.
At the bottom of the table, the percentage of time steps in which a certain number of stars was removed is shown. The different algorithms remove stars from the calculation during a similar amount of time steps. Stars are clearly removed more often when the error on the outlier is higher.

Robustness: Conclusion

The results of the robustness tests show that the Distance method for both AIM and the state of the art methods reliably detect outliers in the data. These outliers greatly reduce the accuracy of the attitude estimation. Using the Distance methods, the outliers can be removed, which increases the accuracy of the attitude estimate significantly. The computational time increases only slightly for AIM when outliers are removed. For QUEST and the SVD method, the computational time increases more drastically. The computational time of

AIM with outlier-remover is still significantly lower than that of SVD and is similar to that of QUEST without outlier-remover.
The attitude estimation algorithms with outlier-remover allow us to obtain reliable star tracker measurements, even when there are outliers in the data. This improvement avoids that the entire attitude estimate is discarded when only a small fraction of the data is unreliable. Because of this, the accurate star tracker can be used during a larger portion of the time, resulting in improved attitude estimation.

5.5.2 Efficiency

In this section, the proposed efficiency improvements are assessed in tests. In these simulations, the effect of reusing database images on the attitude estimation accuracy and computational time is verified. A difference is made between a pointing maneuver, a slow slew maneuver rotating 0.1 degrees per second around the yaw axis, and a slew maneuver rotating 1 degree per second around the yaw axis. These maneuvers were simulated as discussed in section 5.4.1. For each of these three maneuvers, 100 different simulations consisting of 1000 time steps were done. The error was determined by taking the root mean square of the errors of each time step. The computational time and number of reused images were averaged.
The effect of reusing the database images on accuracy and speed was verified by performing these simulations with different values for $ErrorLimit$. This $ErrorLimit$ is the maximum value the quality check may obtain. When $ErrorLimit$ is chosen to be high, the attitude estimation error is allowed to be higher and the procedure will reuse database images more frequently. For a lower $ErrorLimit$, the attitude estimation error needs to remain lower and the database image will be reused less often. When $ErrorLimit$ is set to zero, no database images are reused.
The results given in Figures 5.7-5.9 are a function of the value of this $ErrorLimit$. The percentage of images that is reused is given in the top graph. The relative attitude estimation error increase compared to the error of QUEST is given in the middle graph, and the relative reduction in computational time compared to QUEST is given in the bottom graph.

Pointing

During a pointing maneuver, the star image seen by the star tracker remains more or less fixed. Because the same stars remain in the field of view, the same database image can easily be reused for a longer time.
From the top graph of figure 5.7, it can be seen that the percentage of reused database images increases when the quality check threshold is chosen higher. For a threshold above 0.18, the attitude estimation procedure reuses the first database image for the entire simulation run. Because the star tracker stays

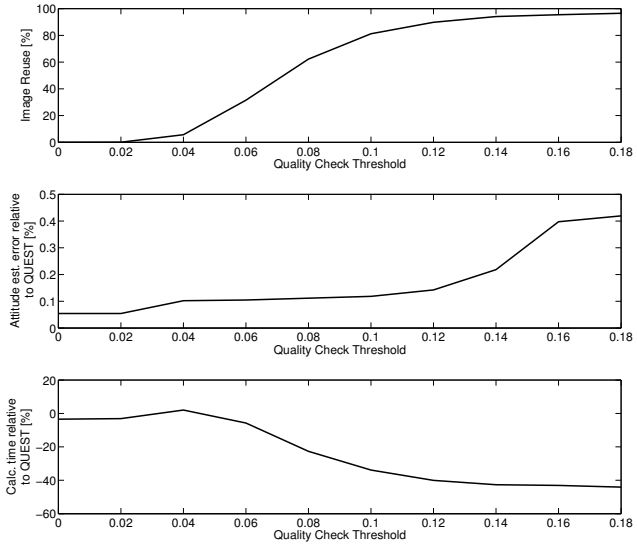


Figure 5.7: Reuse frequency, error increase and computational time decrease during a pointing maneuver

more or less fixed on the same database image, the difference between database-quaternion q_{dat} and the true quaternion q_t remains small. The increase in attitude estimation error therefore remains small, as can be seen from figure 5.3. Even when the same database image is reused during the entire procedure, the error increases only with around 0.4% compared to QUEST.
Because the coordinate conversion step can be discarded, the procedure becomes more efficient when database images can be reused. When no new database images are created, the computational time of the entire procedure is more than 40% lower than that of the procedure using QUEST.
It is interesting to note that there is an initial increase in computational cost. This is explained as follows. When the previous quality index is smaller than $ErrorLimit$, the algorithm reuses the database image. In between the last and the current time step, the attitude has changed, which can result in the fact that the current quality index is higher than $ErrorLimit$. In this case, the algorithm recalculates the attitude using a new database image. As a consequence, the attitude estimation is calculated twice. The computational time therefore increases.

Slew

During a slew maneuver, the attitude of the satellite shifts away from the attitude at which a camera image was taken. From figure 5.3, it follows that this leads to an increase in the attitude estimation error. The larger the distance between the database quaternion q_{dat} and the true quaternion q_t, the larger the error will become.

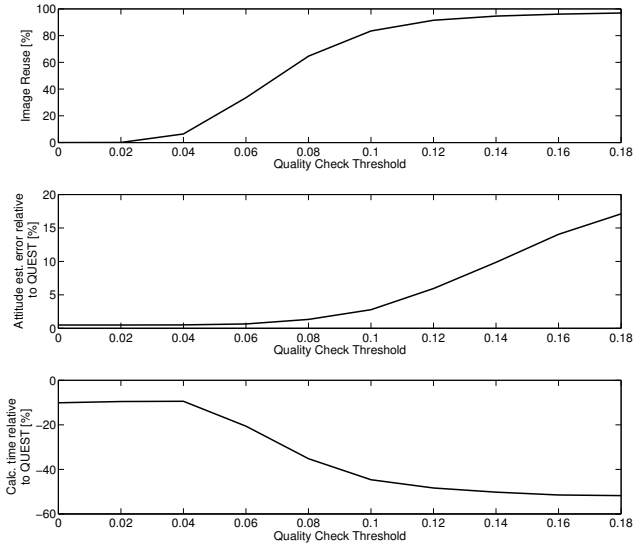


Figure 5.8: Reuse frequency, error increase and computational time decrease during a slow slew maneuver

For the slower slew maneuver, we see in figure 5.8 that the frequency of reusing database images increases up to around 100% when the quality check threshold increases. As opposed to the pointing maneuver, the attitude estimation error increases significantly. When images are reused during 50% of the time, the attitude error is around 1% higher than that of QUEST. When the reuse frequency is around 100%, the error is more than 15% higher than that of QUEST.
The computational time again decreases drastically when images are reused. When images are reused half of the time, the computational time is around 25% lower than that of QUEST. For the maximum reuse of database images, the computational time drops to around half that of QUEST.

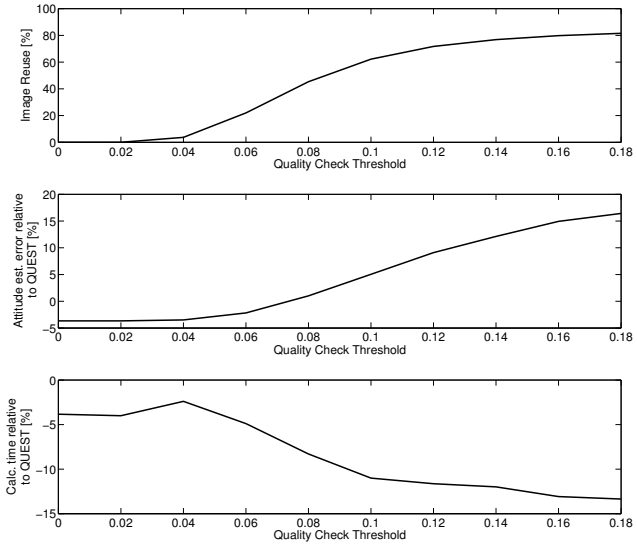


Figure 5.9: Reuse frequency, error increase and computational time decrease during a fast slew maneuver

When the satellite performs a fast slew maneuver, the same database image can not be reused during the entire procedure, as can be seen in figure 5.9. This is because too many of the original stars have left the field of view at the end of the procedure.
The attitude estimation error again increases significantly when more database images are reused. When images are reused during around 80% of the time, the error increases by more than 15%.
The computational time does not drop as much as before, because images cannot be reused as frequently. For a similar attitude estimation accuracy, the reduction in computational time is around 5%. When the maximum amount of database images are reused, the computational time is reduced by a little less than 15%. However, this comes at the cost of an increase in attitude estimation error of around 15%. The initial increase in computational time is also clearly visible here.

Efficiency: Conclusion

The simulations with pointing maneuvers show that reusing database images can greatly increase the speed of the attitude estimation procedure. The computational cost of the entire procedure goes down to almost half of the

computational time needed for the procedure using QUEST. During a pointing maneuver, reusing database images has a very limited impact on the attitude estimation accuracy. The estimation error rises with a few tenths of a percentage. For slew maneuvers, the computational cost is also lowered when more database images are reused. For the slow slew example, the cost can be reduced by 50%. For fast slews, the reduction is smaller, since stars from the previous database images more rapidly disappear from the star image. Reusing the database images during slew maneuvers however significantly increases the attitude estimation error. The error rises in the order of a couple of percents to around 20 percent in simulations. During slew maneuvers, it is therefore less beneficial to reuse database images.
A good approach could be to adapt the quality check threshold when the spacecraft is in pointing or slew mode. The current mode can easily be detected by the ADCS itself by monitoring the rotational speeds. The threshold can be set to zero during slew maneuvers (so that no database images are reused) and to a non-zero threshold during pointing maneuvers. This threshold can be selected by making a trade-off between increased speed and reduced accuracy.

5.6 Conclusion

In this paper, novel methods were discussed to improve the robustness and decrease the computational cost of the star tracker attitude estimation procedure using the AIM (Attitude Estimation using optimal Image Matching) algorithm. AIM has a different approach than state of the art attitude estimation algorithms because it uses focal plane coordinates instead of unit vectors to estimate the attitude. This difference allows us to implement some improvements that significantly increase the already high robustness of the attitude estimation procedure using the AIM algorithm and significantly reduce its already low computational cost.
The robustness of the attitude estimation procedure is increased by introducing an efficient and reliable method to detect and remove outliers. These outliers are stars of which the centroid is determined with a substantially increased positional error. The presence of these outliers in the star data greatly increases the attitude estimation error and makes the star tracker measurements less reliable. The existing TASTE test can detect whether or not such an outlier is present in the data, but cannot pinpoint which of the stars is the outlier. A new method, referred to as the Distance method, is introduced in this paper. This method can detect outliers and also flags which of the stars are the outliers. An efficient approach is then proposed to remove this outlier and recalculate the attitude. While the Distance method and subsequent removal of outliers can be applied to both AIM and the state of the art methods, the approach has a significantly lower computational cost when using the AIM algorithm. This

robustness improvement allows us to obtain accurate star tracker measurements, even when outliers are present, without greatly increasing the computational cost.

The computational cost of the attitude estimation procedure can be reduced by reusing previously calculated database images. Because AIM uses focal plane coordinates to estimate the attitude, the database star coordinates need to be converted. This is as opposed to the state of the art algorithms where the camera image star coordinates are converted. Because the database coordinate values do not change in each time step, these converted coordinates can be reused. When this is done, the computationally complex coordinate conversion step can be discarded, resulting in a reduction of the computational cost. Simulations show that when the database images are reused maximally, the computational cost of the entire procedure is reduced significantly. Reusing the database images comes at the cost of an increased attitude estimation error. During pointing maneuvers, this increase is limited to a few tenths of a percent. For slew maneuvers, the increase is significantly higher. Reusing database images is therefore especially beneficial during pointing maneuvers. The reduction in computational cost can be used to implement a centroiding algorithm with higher performance or to track more stars. It can also be used to increase the update frequency of the star tracker. All of these changes can greatly improve the performance of the attitude estimation of the spacecraft.

The improvements proposed in this paper can lead to a more reliable and less computationally complex star tracker. This can also make the star tracker a more accessible sensor for the growing market of small satellites.

Chapter 6

Development of the KUL ADCS for CubeSats

This section discusses the development of the KUL ADCS for CubeSats. The KUL ADCS is an attitude determination and control system designed and built for small satellites, specifically for CubeSats. In the first part of this section, the novel and rapidly growing field of CubeSats is described in more detail. The pointing requirements and the state of the art ADCS solutions are presented. After this, an overview is given of the KUL ADCS. Its subsystems are described and the performance analysis is presented.

6.1 CubeSats

The last decade, there has been a trend to launch smaller satellites. Recently, there have been several launches of picosatellites ($<$ 1kg), nanosatellites (1-10 kg) and microsatellites (10-100 kg). CubeSats are a subset of nanosatellites that adhere to the "CubeSat Standard" and in general follow similar design rules. Because of the low cost and relatively low complexity of these satellites, the number of CubeSat missions is rapidly increasing.

6.1.1 CubeSat Standard

The CubeSat Standard [117] was first proposed in 1999 by Stanford University and California Polytechnic State University (Cal Poly) [58]. The goal was to promote space research and to give students the opportunity to be part of the design, build and test process of a space mission.
CubeSats are built out of cubic units, hence the name. A standard 1 Unit (1U) CubeSat measures 10x10x10cm, weighs around 1kg and consumes around 1

Watt. Recently, the number of 2U and 3U CubeSats, which consist out of 2 or 3 of the standard Cubes stacked on top of each other, are becoming more common than the 1 Unit CubeSats. These larger CubeSat versions greatly increase the capabilities of the CubeSat. In recent years, 6U and 12U CubeSats have also been proposed.

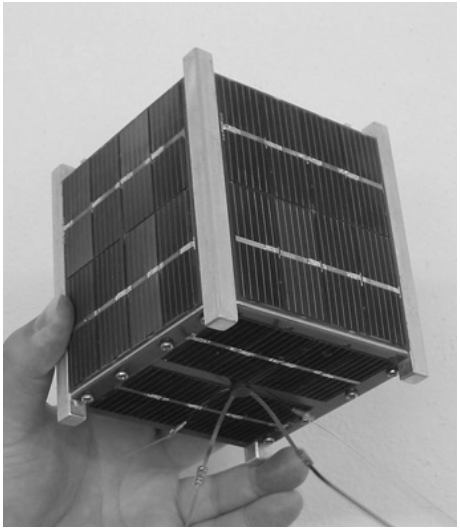


Figure 6.1: A 1 Unit CubeSat. [104]

The main advantage of this CubeSat form factor, which dictates that all 1U-3U CubeSats have the same footprint of 10x10 cm, is that all CubeSats can be launched from the same type of deployer. A standardized deployment system such as the Poly-PicoSatellite Orbital Deployer (P-POD) [69], makes it easier for launch providers to incorporate a CubeSat in their launch vehicle. They know what volume and weight will be taken and the CubeSat standard dictates design rules that help to ensure the safety of the launcher and other satellites from CubeSat failure during launch. Because of the small size and low weight, it is common to see a CubeSat or several CubeSats being launched as a secondary payload, the so called "piggyback ride" [121]. The excess space and mass budget of the launcher is filled and the CubeSat team typically pays a reduced launch cost.
Another advantage of the standard is that technologies developed for one CubeSat can more easily be reused in another CubeSat. This way, CubeSat subsystems and even entire CubeSat busses can be purchased off-the-shelf. This allows CubeSat teams to focus on the specific subsystem or payload they want to develop, while not having to invest a lot of time and effort

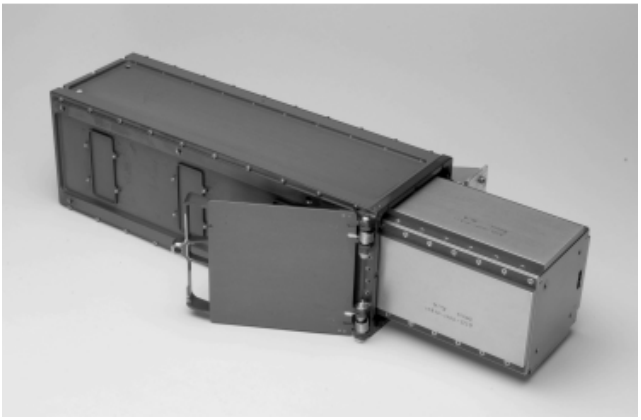


Figure 6.2: A P-Pod with a CubeSat in it [4].

in designing and building the other components. This approach has led to the foundation of several companies that provide CubeSat components and integration services [20, 65, 52, 64]. As a result, the development time and cost is reduced significantly.

Because of the lower cost and lead time of development and launch, CubeSats are very well suited for universities, research institutes and even smaller companies. As a result, the field of CubeSats (and other small satellites) has grown greatly in recent years.

6.1.2 A Growing Field of Satellites

The last decade, there has been a clear trend towards smaller size spacecraft. There are several reasons to explain this recent tendency. The reduced cost and development time is a clear advantage of small satellites over their bigger counterparts. The budgets that are required for space missions are no longer reserved for big governments, also universities and smaller companies can launch a small satellite.

Thanks to the miniaturization of technology, these small satellites are also becoming increasingly powerful and the range of missions they can perform broadens. Missions that would typically be performed by a larger satellite can now be done with (a cluster of) small satellites. The additional advantage of having several small satellites is that there is a risk reduction, since a failure of a single satellite no longer means a failure of the entire mission. Also, a cluster of satellites assures broader coverage for Earth observation or communication

missions.

While CubeSats were originally intended as university projects, their potential has been identified by space agencies such as ESA and NASA [21, 2] and they have set up their own CubeSat or Small Satellite missions [116, 132]. Recently, the economical potential has also been discovered. Several market studies concerning the field of CubeSats and small satellites (<50 kg) have been published [32, 15, 18, 14], with the common conclusion that the field of Small Satellites, and CubeSats in particular will continue to grow with more than 40% per year in the near future.

A recent study by SpaceWorks Enterprises [15] projects between 2000 and 2750 satellites under 50kg to be launched between 2014 and 2020 (in 2014, there were 158 [14]). The vast majority of these would be satellites under 10kg. Another significant trend is the shift from university satellites (70% in 2009-2013 and 35% in 2014-2016) to commercial satellites (8% in 2009-2013 and 55% in 2014-2016). Linked to this trend is the shift from a majority of technology demonstration satellites to a majority of the more commercial Earth Observation and Remote Sensing.

While these numbers are only projections, the rapid growth of the small satellite field is clear. With this expansion, there are bound to be growing pains, conflicts of interest and strain in the market. Without being exhaustive, a number of these can be identified:

- CubeSats typically use amateur radio frequencies in stead of the more expensive commercial frequencies. With the rise of CubeSats, the demand for frequencies rises as well. The International Amateur Radio Union (IARU) will most likely become more reluctant to allocate frequencies to CubeSat teams in the future, forcing them to pay for commercial licenses.
- Launch opportunities will have to be expanded and solutions to provide small satellite launches are currently being investigated [46, 35]. In a launcher market with limited supply and great demand, the commercial CubeSat teams, backed by venture capital, could make it harder for university teams to procure a launch.
- A large number of satellites in Low Earth Orbit make the question of space debris even more urgent. On the other side, CubeSats could be used to clean orbits from dangerous debris [100].

6.2 CubeSat ADCS

This section will look at the ADCS needs of CubeSats and will review the state of the art of attitude determination and control systems for CubeSats. Finally, there is a review of the state of the art for small satellite star trackers specifically.

6.2.1 CubeSat ADCS Needs

Depending on their mission, CubeSats have widely varying ADCS needs. Some CubeSats, and particularly the first ones, have no ADCS at all and can perform their mission regardless of their attitude. The reduction in complexity, mass, volume and power consumption is a big advantage and cost-cutter.

Many CubeSats only require coarse pointing accuracy and knowledge accuracy, in the range of 5 to 10 degrees. This accuracy is sufficient for communication applications (pointing an antenna to a ground station e.g.) or to keep certain instruments in the same general direction. This accuracy can be obtained with simple sensors such as photodiodes, magnetometers and gyroscopes and with magnetorquers only [125].

In order to fly more complex missions such as Earth observation or astronomical missions, the pointing accuracy and knowledge need to be in the range of 1 deg. and lower. This accuracy can no longer be obtained with simple sensors and magnetorquers alone. The attitude determination and control systems that provide this higher accuracy typically add in reaction wheels for improved control. The more accurate pointing knowledge is obtained by using fine Sun sensors, Earth Sensors or a star tracker.

The trend in pointing accuracy for small satellites is reviewed in a recent NASA report [92]. The increasingly complex missions that are being flown by CubeSats are driving an improvement in pointing accuracy (Figure 6.3). The pointing accuracy of nano/pico satellites is improving from around 10 deg. to around 1 deg.

In the future, the ADCS needs of CubeSats will increase. Small satellites will be used more often for mission that require high pointing performance, such as Earth observation and astronomical missions [15]. Future missions that make use of formation flying using differential drag [31] - in which the surface that is perpendicular to the speed direction is changed to control the orbit of the spacecraft - will impose new requirements for agile control. Interplanetary satellites [101] will limit the use of certain sensors (Earth sensor, magnetometer)

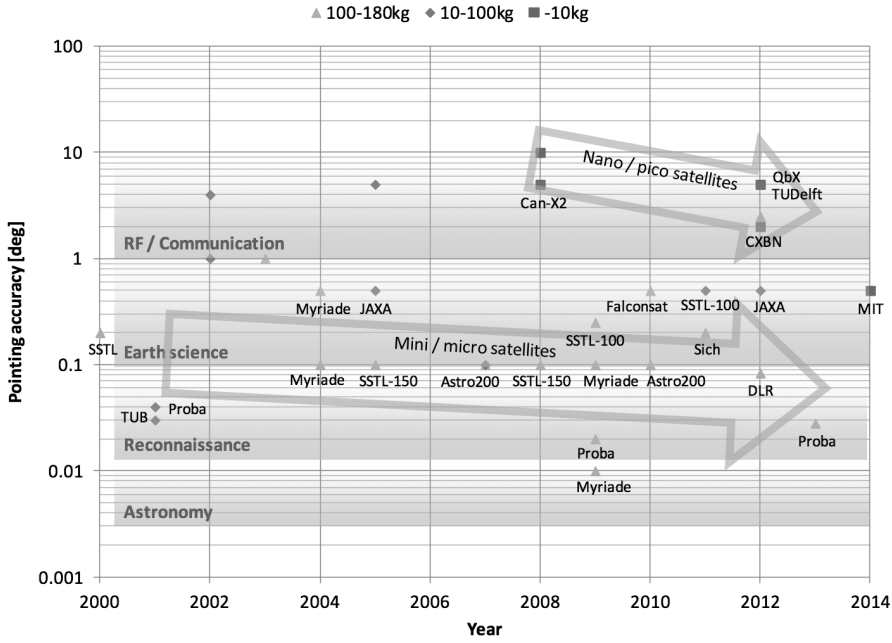


Figure 6.3: The trend in pointing accuracy for satellites <180 kg [92].

and will have to depend more heavily on a star tracker. CubeSats flying in formation or even docking [137, 126] will require highly accurate and agile attitude determination and control, combined with dedicated methods for close-range control.

Apart from improving the ADCS performance, other solutions to accurately point a payload can also be investigated, such as a platform controlled by piezo-actuators that can aim the payload with higher accuracy within the spacecraft[118, 11].

6.2.2 CubeSat ADCS State of the Art

Given the wide variety of CubeSats missions and their corresponding ADCS needs, there is a wide range of CubeSat attitude determination and control systems. The goal of the KUL ADCS project is to make an attitude determination and control system that can deliver sub-degree attitude determination and control. We will therefore mainly review the state of the art of systems that can deliver similar performance. As mentioned above, several systems exist that deliver lower performance with lower component complexity,

but these systems will not be reviewed.

High pointing accuracy can be obtained with reaction wheels or thrusters. Thruster-based systems are very novel in CubeSats [70, 90] and are still waiting for their mainstream breakthrough. The state of the art in fine control is obtained with reaction wheels. To obtain high accuracy, fine sun sensors, earth sensors or star trackers can be used. Important to note is that sun sensors and earth sensors (that work with visible light) can not determine the attitude in eclipse.

Table 6.1: An overview of the state of the art in fine attitude determination and control systems for CubeSats.

ADCS	Developer	RW T. (mNm)	RW M. ($mNms$)	Dim. (mm^3)	Mass (g)	Star Tra.
MAI-100 [80]	Maryland Aerospace	0.635	1.1	100x100x79	865	No
MAI-200 [81]	Maryland Aerospace	0.635	9.35	100x100x79	907	No
MAI-400 [82]	Maryland Aerospace	0.635	9.35	100x100x50	694	No
XACT [12]	Blue Canyon Tech	6	15	100x100x50	850	Yes
XACT Lite [13]	Blue Canyon Tech	6	15	100x100x50	700	No
Cube 1 [126] [130]	Stellenbosch Surrey	0.23	1.7	96x90x60	400	No
Cube 3 [122]	Stellenbosch Surrey	0.23	1.7	96x90x60	460	No
iADCS-100 [10]	Berlin Space Tech	0.087	1.5	96x90x32	250	Yes
KUL ADCS	KU Leuven	6	12	96x90x80	985	Yes

Table 6.1 gives the main performance values for the current state of the art systems that can deliver pointing performance in the range of 1 degree. The final pointing performance of these systems depends on the satellite, orbit and other factors and is therefore not added in the table.

Reaction wheels delivering higher torque (RW T.) allow for more agile control. A larger momentum (RW M.) assures a longer functioning of the reaction wheels

before their momentum needs to be desaturated by an external torque, often generated by magnetorquers. The systems are divided in two groups based on their dimension footprint. Systems with a footprint of 100x100mm have the same footprint as a CubeSat and replace part of the CubeSat structure. Systems with a 96x90 (PC104) footprint can be placed on the electronics stack. The second option allows for easier integration in a standard CubeSat structure and allows to more easily integrate solar panels on that structure. The weight is mainly dictated by the reaction wheels and systems with high-momentum reaction wheels will therefore have a larger weight.

6.2.3 CubeSat Star Tracker State of the Art

One of the bottlenecks that limits high pointing performance for CubeSats is the lack of highly accurate attitude determination sensors. A CubeSat star tracker could solve this issue, but its development is complex. A star tracker consists of a small telescope, an optical baffle, a detector and a processing unit to match the stars in the focal plane of the star tracker with a catalogue of known star positions. To reduce the volume, mass and power footprint to levels that are acceptable for CubeSats while maintaining accuracy, changes in these four areas are required.

A recent NASA study reviewing the technologies for small satellite technologies reviews the state of the art for CubeSat star trackers. The main conclusion is that some star trackers have been designed and built, but their accuracy is not entirely in the range of that of other (larger) star trackers, as can be seen in Figure 6.4. More importantly, the Technology Readiness Level (TRL) of these systems is still quite low (TRL ≤ 6). This means that these systems have not yet shown good performance in-orbit.

A compact, highly accurate star tracker could fill a gap in the current state of the art. Such a sensor could greatly improve the attitude determination and control performance and would allow CubeSats to perform more complex missions. The algorithms that were developed in this work facilitate the development of such a star tracker.

6.3 The KUL ADCS Mission

The KUL ADCS is designed to deliver the performance needed in CubeSat missions requiring high attitude determination and control performance. The mission that will implement the KUL ADCS first is the SIMBA CubeSat mission. The Sun-earth IMBAlance (SIMBA) satellite by the Royal Meteorological Institute of Belgium (RMIB) is an In Orbit Demonstration satellite. The Simba

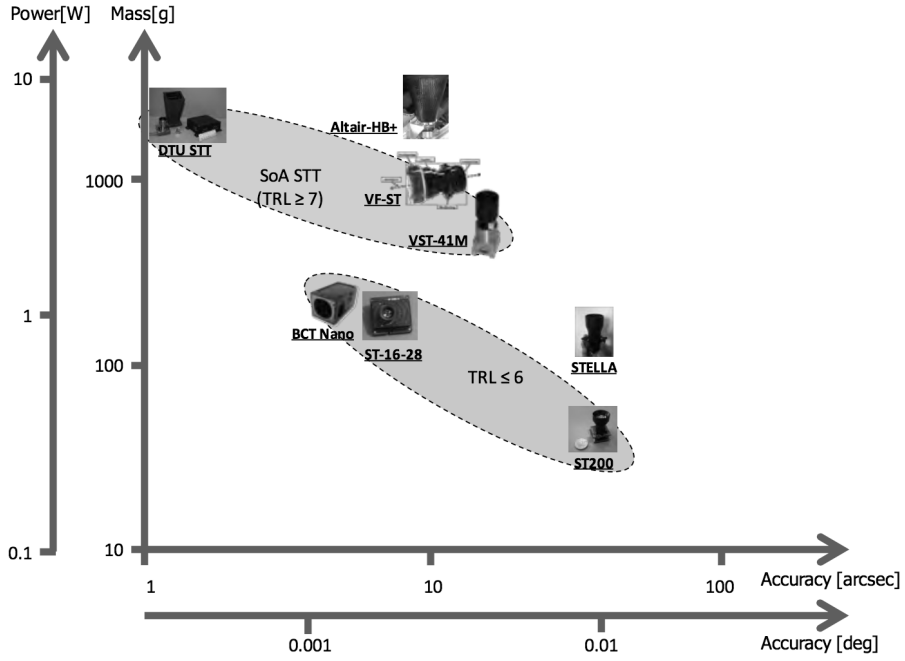


Figure 6.4: An overview of the state of the art of star trackers for small satellites and CubeSats [92].

scientific payload objectives are to make maps of the emitted thermal radiation and reflected solar radiation of the Earth, and to make with the same instrument measurements of the incoming solar radiation. Thus Simba will contribute to the monitoring of the Essential Climate Variables of total solar irradiance and the earth radiation budget. As a longer term objective, this should lead to the measurement of the Earth radiation imbalance [34].

6.3.1 Mission ADCS Requirements

SIMBA is an Earth observation satellite and will nominally be in nadir pointing mode (towards the center of Earth). At intervals of around three months, the CubeSat will be turned around into zenith pointing mode (away from the center of Earth) for a short period to calibrate the instruments.

The ADCS requirements are dictated by the needs of the payload instrument. To measure the radiation of the Earth, the payload needs to have the entire Earth within its field of view. It is the task of the ADCS to maintain the

attitude so that this can be done. The scientific return improves when the attitude is more stable and better known.

The driving factor for the pointing performance is the cavity radiometer. From the orbit parameters and the field of view of the instrument, it follows that the pointing accuracy and pointing knowledge should be better than 3.5 degrees to have a successful mission. It has been established that with an accuracy below 1 degree, the scientific return improves significantly. These numbers rule out a system without reaction wheels and require an accurate set of sensors.

Since the satellite is also a technology demonstrator, it should also be possible to test it flexibly. That is why there was a mode implemented to point to a user-defined attitude so that other pointing operations or slews can be performed and validated.

Apart from this performance in the CubeSats nominal mode, the ADCS needs to be able to detumble the satellite. Upon release from the launcher, CubeSats often have high rotational rates (up to 30 deg/s have been recorded) [97]. To recover the CubeSat from these spin rates, a detumbling control law (B-dot law) using magnetorquers is implemented [44].

The different modes that make up the entire ADCS are depicted in Figure 6.5. The figure also shows the possible transitions between modes.

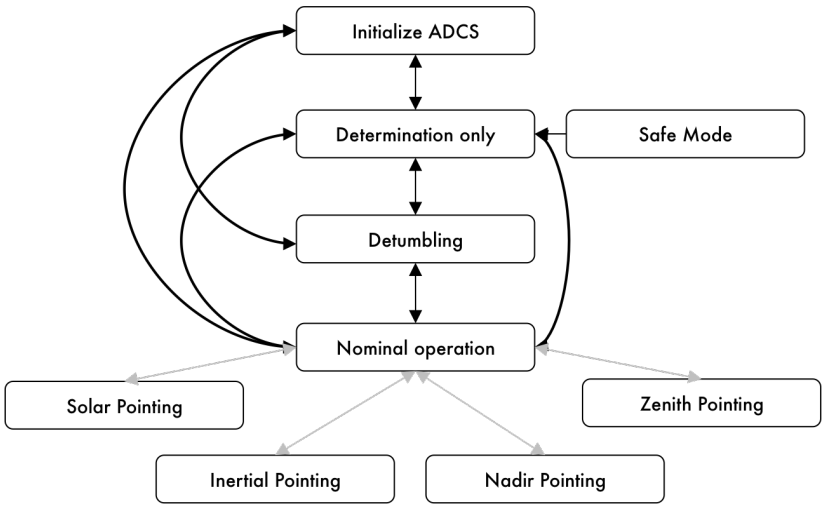


Figure 6.5: The modes of the KUL ADCS and transitions between them.

6.4 Overview of the KUL ADCS

This section will give a concise overview of the KUL ADCS and will describe the subsystems of the attitude determination and control system.

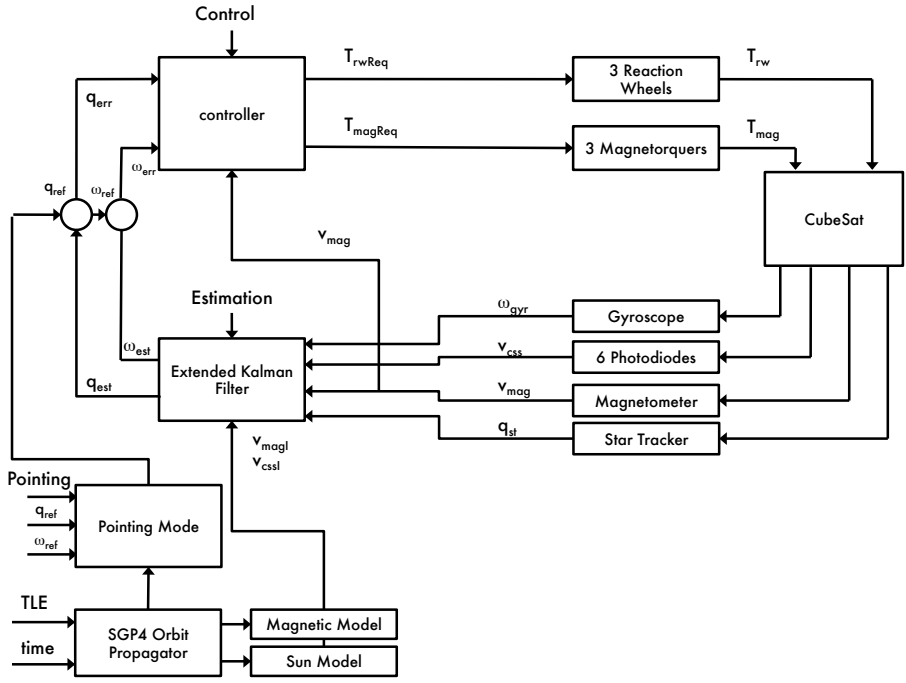


Figure 6.6: A Functional Diagram of the KUL ADCS.

In essence, the KUL ADCS is a typical feedback loop. A set of sensors measures the attitude and rotational rate of the satellite. The state estimator uses this information, combined with a model of the satellite kinematics to estimate the attitude and rotational rate. The controller determines the action that needs to be taken to get to the desired attitude and rotational rate and the actuators are used to control the attitude. This is schematically displayed in the top part of Figure 6.6.

The magnetometer and sun sensor determine the attitude by comparing their measured vectors in a body frame of the CubeSat to the same vectors (towards the sun and along the magnetic field) in an inertial frame. For these sensors to give meaningful information, the magnetic field direction and position of the sun in an Earth Centered Inertial frame need to be known. This information is

obtained from knowledge of the position of the satellite in its orbit. The orbit determination segment and the models for the sun location and the magnetic field, together with the algorithms that calculate the desired attitude and rotational rate from the selected pointing mode, constitutes the lower part of Figure 6.6.

The different components are shown in the two views of the KUL ADCS in Figure 6.7. The actuators of the ADCS are placed inside a container. In this figure, the top container is removed so that the components inside can be seen. Three PCBs are used in the ADCS. The camera PCB holds the star tracker imager, the ADCSDRV PCB holds the drivers for the actuators and the ADCSCPU holds the remaining electronics, the main processor and the gyroscope.

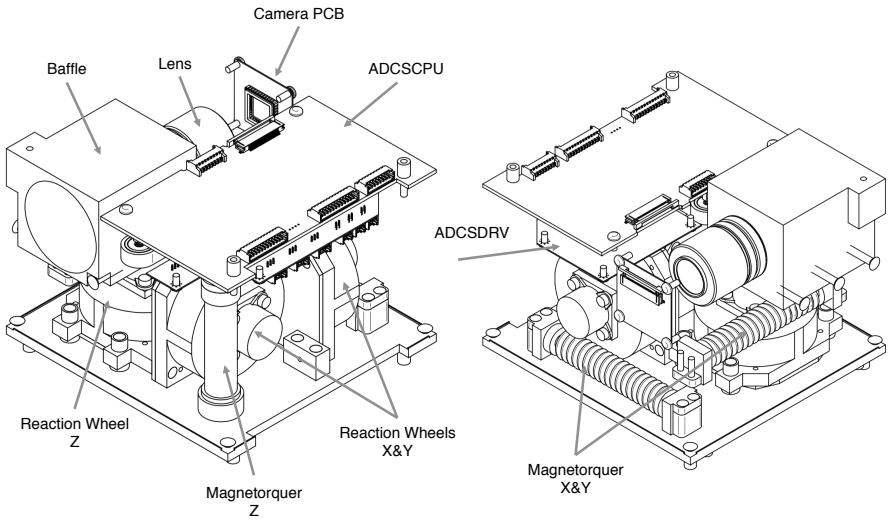


Figure 6.7: Two views of the KUL ADCS without the top container, showing the components of the KUL ADCS

The ADCS container structure is used to conveniently mount the ADCS in the CubeSat structure as can be seen in Figure 6.8. The ADCS can be mounted like a PC104 stack which is typically used in CubeSats. This facilitates the design and assembly of the CubeSat.

6.4.1 Actuators

The KUL ADCS has two main sets of actuators: 3 reaction wheels and 3 magnetorquers. The reaction wheels offer accurate and agile control. The

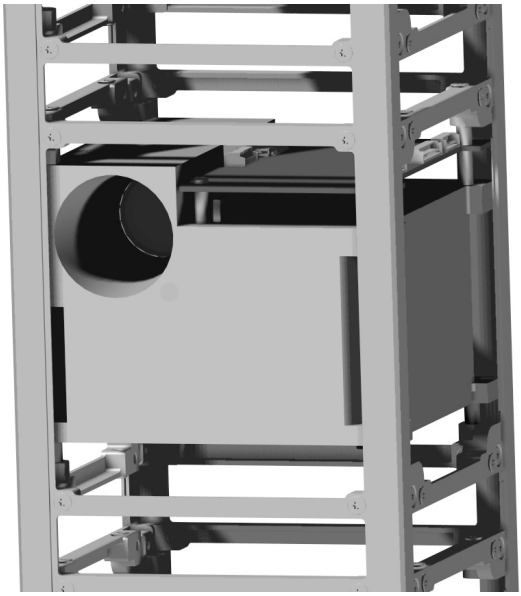


Figure 6.8: The KUL ADCS in the CubeSat structure

magnetorquers are used to detumble the satellite, desaturate the reaction wheels and to have a back-up coarse pointing mode.

Reaction Wheels

Reaction wheels are actuators that allow to control the attitude of a satellite by changing the distribution of momentum inside the satellite [115]. The total inertial momentum of the satellite is not changed by a reaction wheel.

A reaction wheel is built up by a flywheel that is mounted on an electrical motor. The motor changes the rotational velocity and hence the angular momentum of the reaction wheel. By the law of conservation of angular momentum, a change of the angular momentum of the reaction wheel will result in a change of angular momentum of the satellite.

The momentum of a reaction wheel can be calculated as in equation 6.1

$$h_{rw} = I_{rw}\omega_{rw}, \tag{6.1}$$

where I_{rw} is the moment of inertia of the flywheel and ω_{rw} is the rotational velocity of the flywheel. To increase the momentum capacity of the reaction wheel, one can increase either the moment of inertia of the flywheel or the

maximum angular velocity of the flywheel. An increase in the moment of inertia comes at the cost of a larger and/or heavier flywheel. The flywheel should be designed to have a high ratio of inertial moment over mass. The angular velocity is bounded by the motor performance and the bearing life time.

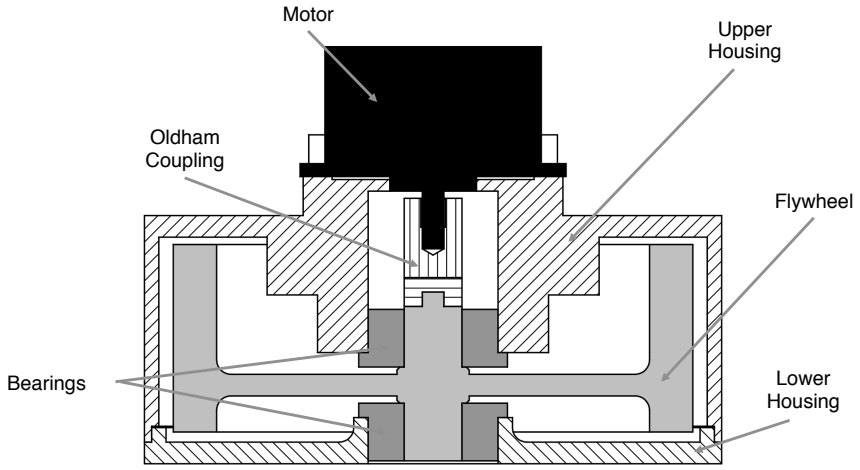


Figure 6.9: A cross section drawing of the Z-reaction wheel.

The KUL reaction wheels have been developed during several Master Thesis projects [53, 128]. The reaction wheels use a brushless DC motor (1509 006B) of Faulhaber [39]. The bearings used are the SSF 684-2Z P5 C4 GPR J G442 CP of GRW [54], which have a specific lubricant that is vacuum-compatible. The mechanical components have been designed in-house. The flywheel is designed to have a high moment of inertia for a low mass, by placing the main mass as far away as possible from the rotational axis. A simple Thompson coupling is used to compensate slight misalignments between the motor rotor and the flywheel. These components are displayed in Figure 6.9. The diameter of the upper housing is 40mm.

All three reaction wheels will be used when the satellite is in fine pointing mode. The accuracy and agility of the reaction wheels will allow the SIMBA spacecraft to reach its pointing requirements. In coarse pointing mode, one reaction wheel will be used together with two magnetorquers to control the spacecraft.

Magnetorquers

A magnetorquer is a coil which is typically wound around a magnetic core. When a current is passed through the coil, a magnetic moment is generated. Magnetorquers control the attitude of a spacecraft by generating a magnetic

moment that interacts with the Earth's magnetic field as is described in equation 6.2:

$$T_{mtq} = M_{mtq} \times B, \tag{6.2}$$

Where T_{mtq} is the torque generated by the magnetorquer, M_{mtq} is the magnetic moment generated and B is the Earth's magnetic field vector [115].

The KUL ADCS has three magnetorquers on board, the X- and Y-magnetorquer are the CubeTorquer built by the Stellenbosch University [123]. The Z-magnetorquer is designed in-house [91], because the dimensions did not allow the use of the CubeTorquer.

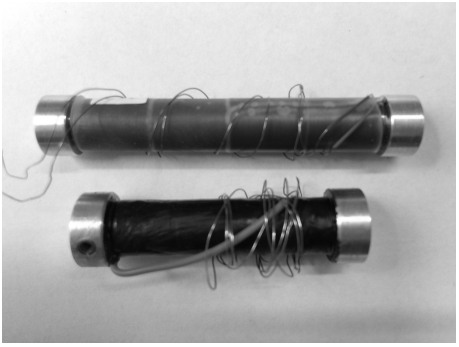


Figure 6.10: A purchased magnetorquer (top) and the in-house developed Z-magnetorquer (bottom).

All three magnetorquers will be used in the detumbling mode of the spacecraft to lower the rotational velocity. In coarse pointing mode, two magnetorquers are used together with a reaction wheel to control the attitude of the spacecraft. When the reaction wheels are used, the magnetorquers will also be used to dump the momentum of the reaction wheels. Under influence of a secular disturbance torque or due to other reasons, the reaction wheels might continuously spin up in one direction. At a certain point, the reaction wheel reaches its maximum rotational velocity and can at this point no longer produce a torque in that direction. To prevent this, the reaction wheel speed is kept at a predetermined level by using a desaturation control law. This control law powers the magnetorquers such that the reaction wheel momentum remains more or less constant.

6.4.2 Sensors

The KUL ADCS uses a set of different sensors to determine the attitude. The star tracker is by far the most accurate sensor. Its advantage is that it can also work in eclipse. The downside is that it is a complex unit and that stray light from the Sun or Earth can prevent it from functioning. Photodiodes and a magnetometer measure vectors towards the sun and in the direction of the Earth's magnetic field respectively. With two vectors, the attitude can be determined. A gyroscope measures the rotational rate of the spacecraft and allows to propagate the attitude.

Star Tracker

The KUL star tracker consists of a baffle, lens, detector and processing unit to run the algorithms.

Baffle The baffle is a cone-shaped structure that is place in front of the lens to keep stray light out. This stray light, coming from bright sources such as the Sun or Earth makes it hard or even impossible for the detector to detect stars. A baffle is optimized to reject the stray light as much as possible. For that purpose, it is covered with a black coating and has reflecting structures called vanes on its inside. A cross section of a prototype (non-coated baffle) can be seen in Figure 6.11.

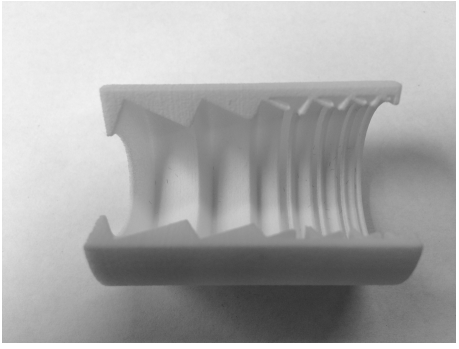


Figure 6.11: A cross section of an uncoated baffle.

A longer baffle rejects more stray light. The star tracker is therefore designed to have a maximum length of the optical axis, to be able to make the baffle as long as possible. To further optimize the rejection of stray light, a master thesis project was set up to optimize the shape, dimensions and placement of the inner vane-structure using Ray Tracing [99]. The designed baffle will be 3D-printed and coated with a black coating.

Lens A simulation was performed to select the lens [23]. Taking into account the sampling time, the number of stars we would like to see on the detector, the allowed dimensions of the system and other design parameters, the 85-355 lens of Edmund Optics was selected [94]. This lens has a focal length of 25mm and a diameter of 26mm.

Detector A CMOS detector is used to detect the stars. The EV76C661 of e2v [136] was selected because of its low noise levels and high sensitivity, also in the near-infrared. The sensor has 1280x1024 pixels, low power consumption (200 mW) and allows to have 60 frames per second at full resolution. The monochrome version of the camera is used because the algorithms do not require color information.

Processing Unit The star tracker algorithms are run on the main processor of the ADCS. This is an ARM Cortex-M4 [76] processor. This processor is sufficiently powerful for the applications that are required and has a lower power consumption. The star tracker for the KUL ADCS uses the algorithms discussed in this work.

Magnetometer

A magnetometer is a sensor that measures a magnetic field. In the KUL ADCS, a 3-axis MEMS magnetometer is used to measure Earth's magnetic field vector. The selected sensor, Honeywell's HMC5883L [61], is small, low-cost and has a low power consumption.

The magnetometer will not be placed within the ADCS structure, but rather in the payload compartment. The reason for this is that the ADCS contains magnetorquers and the permanent magnets in the reaction wheel motors. These disturb the measurements of the magnetometer. In the payload compartment, the magnetometer is further away from these disturbances and their effect is diminished. Even with this larger distance, the magnetometer will not measure when the magnetorquers are powered. The magnetometer and magnetorquers will work in a 20(magnetometer)/80(magnetorquers) duty cycle.

Photodiodes

A photodiode is a sensor that converts light into current. Six SLCD-61N8 photodiodes of Silonex Inc. [63] are placed on the six sides of the CubeSat. Based on the measured current from each of these sensors, the vector towards the sun can be calculated.

Gyroscope

Gyroscopes measure the angular rate of the spacecraft. In the KUL ADCS a 3-axis MEMS gyroscope is used: the ITG-3200 of Invensense [62]. The noise characteristics of this sensor were analysed using an Allan Variance [68].

6.4.3 Estimator

The spacecraft attitude and angular rate is measured by a set of different sensors. All of these measurements are subject to noise. An estimator is used to determine the attitude, angular rate and the bias of the gyroscope out of these measurements. In the KUL ADCS, an Extended Kalman Filter is used for this purpose [68].

The Extended Kalman Filter uses a model of the system and of the uncertainties to recursively estimate the unknown variables. The filter works in two steps: First it propagates the spacecraft attitude and covariances using the rotational rate of the gyroscope. In the second step, the measurements of the sensors are added to update the states and covariances. The advantage of the Extended Kalman Filter discussed in [68] is that the sensor measurements can be added flexibly, i.e. not all sensor measurements need to be available to update the states. Since photodiode measurements are not relevant in eclipse, star tracker measurements are not always available and the magnetometer works in a duty cycle with the magnetorquers, this is a necessary feature.

6.4.4 Controller

The ADCS has three main control laws.

Detumbling

After the CubeSat has been released from the spacecraft, it may have considerably large angular rates. The angular rates are reduced during the detumbling phase. To detumble the CubeSat, a B-dot controller will be used. This is a control law that relies on the three magnetorquers as control actuators. The control law is based on the measurements of the change of magnetometer signals [44].

Coarse Control

In coarse control mode, the Y-reaction wheel is used together with the X- and Z-magnetorquer. By doing this, the reliance on the more failure-prone reaction wheels is lower than in Fine control, while there is still full control over all three axes (which is not the case when three magnetorquers are used alone). The control equation described in [115] was implemented.

Fine Control

The fine control mode of the KUL ADCS uses a PD controller to control the three reaction wheels separately.

6.4.5 Orbit Determination

Knowledge of the orbit position is necessary for a number of reasons. To extract meaningful information from the magnetometer and photodiodes, the vectors towards the sun and along the magnetic field need to be known in the inertial frame. These vectors are calculated from the current position of the satellite and the current time. Furthermore, the position knowledge is required to determine the quaternion in which the satellite is Nadir, Zenith and Sun pointing, since we need to know the position of the Earth or Sun relative to the spacecraft.

The KUL ADCS does not comprise an orbit sensor such as a GPS and therefore relies on Two-line Element Sets (TLEs) and an orbit propagator to estimate the orbit position. At frequent intervals, such as once per day, the CubeSat will receive a TLE from the ground station. A TLE consists of two lines of 80-column ASCII text containing the main orbit parameters and the position and velocity at a certain time. This information is made available by the United States Air Force which tracks space objects.

An SGP-4 orbit propagator is then used to propagate the orbit. This propagator takes into account effects such as drag, perturbations because of the Earth's not perfectly spherical shape, gravitation effects from the Sun and Moon, etc. The SGP4 model has an error which grows at around 2km per day [127]. This model is implemented in the KUL ADCS.

Magnetic Field Model

The KUL ADCS uses the IGRF-11 Model to model the Earth's Magnetic field [43]. This model is revised every five years by the International Association of Geomagnetism and Aeronomy and allows to calculate the intensity and direction of the Earth's magnetic field vector at a given location.

Sun Model

The KUL ADCS uses a high accuracy algorithm to calculate the position of the sun presented in [85]. The error on the inertial sun vector is below 1 arc second.

6.5 Performance Analysis

The performance of the ADCS was simulated in a Matlab/Simulink environment. The simulated pointing performance plays an important role in the mission design of the SIMBA CubeSat. Based on these simulations values, the expected scientific value of the CubeSat can be estimated.
The simulation was done for four different cases:

- Coarse Estimation with Coarse Control

- Coarse Estimation with Fine Control
- Fine Estimation with Coarse Control
- Fine Estimation with Fine Control

Coarse control uses two magnetorquers and 1 reaction wheel and fine control uses all three magnetorquers and all three reaction wheels. Fine Estimation is with use of the star tracker, coarse estimation is without the star tracker and only with gyroscope, magnetometer and photodiodes.

6.5.1 Simulation Setup

The simulator was programmed in Simulink. For each simulation case, 5 test sequences of 20.000 seconds (around 4 orbits) were run. The results are obtained by looking at the final half of each simulation, to make sure that the effects of the step maneuver have disappeared and we are looking at the pointing performance. The simulator has the following characteristics:

CubeSat Modelling

- Attitude Kinematics and Dynamics Propagator
- Initial rotational rate: (2, 2, 2) $\frac{deg}{s}$
- Step maneuver of on average 30 deg around each axis
- $I_{xx} = 0.005 kgm^2$, $I_{yy} = 0.025 kgm^2$, $I_{zz} = 0.025 kgm^2$
- SGP4 Orbit propagator
- IGRF-11 magnetic field model
- Sun vector model
- Disturbance torques as described in appendix B

Actuator Modelling

- Reaction Wheels:
  - Electrical Limits
  - Friction
  - Torque Disturbance
- Magnetorquer
  - Limited magnetic moment
  - Disturbance magnetic moment with stdv 5% of maximum moment
  - Duty cycle with magnetometer (0.8 MTQ, 0.2 MTM)

Sensor Modelling

- Gyroscope
  - Initial bias: $0.001 \frac{deg}{s}$
  - ARW: $0.012 \frac{deg}{s^{\frac{1}{2}}}$
  - RRW : $0.0006 \frac{deg}{s^{\frac{3}{2}}}$
- Sun sensor
  - Measured vector with standard deviation of 5deg around each axis
  - Does not function during eclipse
- Magnetometer
  - Measured vector with standard deviation of 3 deg around each axis
  - Duty cycle with magnetorquers (0.8 MTQ, 0.2 MTM)
- Star Tracker
  - Quaternion with expected standard deviation on each axis. In these simulations, when used, the star tracker is continuously used.

6.5.2 Simulation Results

To analyse the performance, two main questions are answered:

- How does the root mean square accuracy compare to the requirement?
- What percentage of the time does the system reach the requirement?

Requirements

The requirements are given in Table 6.2. The root mean square pointing accuracy during daytime and eclipse should be below 3.5 degrees. The root mean square pointing knowledge should be below 1 degree during daytime and eclipse. Because an upper limit of 3.5 degrees pointing accuracy and 1 degree pointing knowledge cannot be met during the entire orbit time by all configurations, requirements were set on the percentage of time the system should meet these levels. The pointing accuracy should be below 3.5 degrees for 80% of the day part of the orbit and during 50% of the eclipse part. A similar requirement is set for the knowledge accuracy.
The results are given in Table 6.3. The cases where the results are within the requirements are highlighted. C stands for Coarse, F for Fine.

Table 6.2: The Requirements for the KUL ADCS in the SIMBA mission.

Requirement	Value
Pointing Accuracy Day	3.5 deg
Pointing Accuracy Eclipse	3.5 deg
Knowledge Accuracy Day	1 deg
Knowledge Accuracy Eclipse	1 deg
% Time Pointing accuracy is reached during Day part	80
% Time Pointing accuracy is reached during Eclipse	50
% Time Knowledge accuracy is reached during Day part	80
% Time Knowledge accuracy is reached during Eclipse	50

Table 6.3: The performance of the KUL ADCS in the SIMBA mission.

Case	Point. acc. Day (deg)	Point. acc. Ecl. (deg)	Know. acc. Day (deg)	Know. acc. Ecl. (deg)
C Es. C Con.	15.3	22.5	**0.7**	20.0
C Es. F Con.	**0.9**	6.1	**0.7**	5.7
F Es. C Con.	**2.4**	**2.7**	**0.04**	**0.04**
F Es. F Con.	**0.13**	**0.13**	**0.04**	**0.04**
Case	% time Point. acc. Reached Day	% time Point. acc. Reached Ecl.	% time Know. acc. Reached Day	% time Know. acc. Reached Ecl.
C Es. C Con.	9	7	69	4
C Es. F Con.	**98**	43	75	7
F Es. C Con.	**82**	**74**	**100**	**100**
F Es. F Con.	**99**	**100**	**100**	**100**

Coarse Estimation and Coarse Control

This mode only meets the knowledge requirement during day. The pointing accuracy during day is around 15 deg. In eclipse the pointing accuracy is reduced to around 20 degrees. In eclipse the pointing knowledge is greatly reduced because the lack of photodiode measurements does not allow a full update of the Extended Kalman Filter.

The required Pointing accuracy during day is reached in about 9 percent of the daytime, the knowledge is reached in about 70 percent of daytime. In eclipse, both pointing and accuracy are met in around 5 percent of the time or less.

Because the satellite drifts away from its desired attitude during eclipse, a significant fraction of the daytime is consumed to reorient the CubeSat. The larger the drift during eclipse, the shorter the time pointing accuracy will be reached during day. As a sidenote, during 40 percent of the daytime, the pointing accuracy is below 10 deg.

Coarse Estimation and Fine Control

This mode suffers similar disadvantages in eclipse as the above case. The difference here is that the percentage of time at which pointing accuracy is reached is significantly larger and the pointing accuracy is a lot better. The reason for this is that the agile reaction wheels allow to compensate the drift (occured during eclipse) much faster. The satellite obtains its desired attitude faster after eclipse and keeps it more steady. During eclipse, the pointing and knowledge performance is also better than the previous case because the reaction wheels do a better job at maintaining the attitude correctly in eclipse.

Fine Estimation and Coarse Control

All of the requirements are met in this mode. During the simulations, the star tracker was continuously used and this is reflected in a very good knowledge accuracy (during daytime and eclipse) and also a high percentage of time during which the knowledge requirements are met.

Because there is no reduced knowledge and no drifting of the CubeSat during eclipse, the pointing accuracy is similar during daytime and eclipse. Because no drift needs to be compensated when the CubeSat exits eclipse, the pointing accuracy is higher and the percentage of time during which pointing is achieved is higher.

Fine Estimation and Fine Control

Again, all the requirements are met. This case is similar to the one above, but the more agile and accurate reaction wheels allow to control the attitude more accurately. This results in a better pointing accuracy and again a continuous achievement of accuracy during daytime and eclipse.

6.6 Conclusion

With the addition of a star tracker and three reaction wheels, the KUL ADCS will deliver high pointing accuracy and agile control. The development is still ongoing, with a functional and environmental test campaign as the next step. Aside from an interesting addition to the field of CubeSat attitude determination and control, this ADCS has been very valuable to give students hands-on experience with space research.

Chapter 7

Conclusion and Future Work

This work presents novel developments in the field of attitude determination and control of satellites. Increasingly complex space missions demand more accurate and agile control of the spacecraft. One of the key technologies to achieve this is the star tracker, a sensor that accurately determines the attitude of the spacecraft. The main part of this dissertation is the development of novel algorithms that improve the performance of this sensor. The final part of this work focusses on the development of a complete attitude determination and control system for the rapidly growing market of small satellites.

7.1 Research Achievements

The main contribution of this work is the development of novel star tracker algorithms. The star tracker cycle consists of three main steps. For each of these steps, a novel algorithm with improved characteristics was developed and was compared in tests to the state of the art. The second contribution of this work is the development of a full attitude determination and control system for a CubeSat.

7.1.1 Novel Star Tracker Algorithms

Star tracker performance can be described by three main characteristics:

- Computational cost: a lower computational cost can lead to a faster update rate of the star tracker. It can also allow the use of cheaper components.
- Accuracy: higher accuracy leads to improved pointing accuracy.

- Robustness: higher robustness improves the availability of the star tracker measurements.

In the following paragraphs, we briefly position the novel algorithms within the state of the art, discuss their functioning and summarize their performance compared to the state of the art.

Centroiding

The centroiding algorithm determines the centroid of each star. Two main approaches can be identified in the state of the art. One group of algorithms determines the centroid by taking a weighted average of the pixel intensities. This group is very computationally efficient, but has lower accuracy. The second group fits a model of the known point spread function (PSF) through the intensity data and yields higher accuracy for a higher computational cost.

The algorithm of chapter 2, which was published in [29], uses the approach of the second group in the sense that it fits a model of the PSF through the data. Because of this, the accuracy is comparable to that of the most accurate algorithms. The main difference with the state of the art algorithms is that the novel algorithm does not use an iterative least squares approach to fit the model. Instead, closed form equations were derived to fit the model. This results in a very computationally efficient algorithm which is more than 80 times faster than the most accurate algorithm in simulations. The simplifications that were made to derive these closed form equations have a limited negative effect on the accuracy, with an error increase in the range of 15% in simulations.

The novel algorithm combines the advantages of the two groups. It is in the range of the fastest algorithms when computational cost is concerned and has accuracy comparable to that of the most accurate algorithms. Furthermore, a hybrid algorithm was presented in which the novel algorithm gives an initial estimate for the iterative least squares approach. The hybrid algorithm yields the same accuracy as the most accurate state of the art algorithm while being significantly faster.

Lost-in-space

The lost-in-space algorithm identifies the stars in the image without a-priori knowledge. The algorithm extracts features from the camera image and searches the entire star database on board of the star tracker to find matching database stars for the detected stars. Lost-in-space algorithms that are currently used on board of star trackers extract features from so-called triplets of stars. Several sets of three stars, a triplet, are selected from the image and features are extracted from it. A wide range of algorithms, based on different features such as the inter-star angle, the area of the enclosed triangle, the angles of the triangle, etc. exist. The features are matched with a preprocessed database

of star triplet features to identify the stars. More recent research focusses on the use of neural networks. The high complexity of these algorithms currently limits the use of this approach.

The algorithm of chapter 3, which was published in [27], is not based on star triplets. The algorithm transforms the camera image using a Shortest Distance transformation. This results in a 2D look-up map that holds the distance to the closest star at each location of the map. The algorithm then generates "database images" out of the star database at evenly spread locations over the night sky. These database images are the images the star tracker would see at a given location in the sky. The images are matched with the camera image and the database star positions can easily be entered into the 2D look-up map. A higher number of database and camera stars that are close together points to a higher resemblance of the images. The database image with highest resemblance is withheld and the stars can be identified based on this image.

The novel approach leads to significantly higher robustness than that of the state of the art algorithms. As an example, the algorithm determines over 99% of camera images correctly when 400 false stars are added, distortions of 300 (1σ) arc seconds are present and the brightest star is missing in the image. Whereas the star triplet approach is rather sensitive to distortions, the novel algorithm is very robust to false stars, distortions in star position and missing stars. On top of this high robustness, the algorithm provides a useful confidence value. The algorithm always returns a result, but based on the percentage of close stars (higher is better), the algorithm can automatically assess the confidence in the result. The computational complexity is in line with that of state of the art algorithms and can be significantly reduced if a coarse estimate of the attitude is available. In that case, the location of generated database images can be focussed around the assumed portion of the night sky.

Tracking

The tracking algorithm finds the transformation between camera and database star coordinates. The calculated angles of rotation determine the attitude of the spacecraft. All state of the art algorithms solve Wahba's problem. They calculate the rotation matrix that minimizes the distance between the camera star vectors and rotated database star vectors. This rotation matrix is then used to determine the attitude of the spacecraft. The state of the art algorithms can be divided into two groups. One group is based on robust algorithms to calculate the eigenvalue problem or singular value decomposition to solve Wahba's problem. These algorithms are robust but slow. The other algorithms use fast iterative solutions and are faster, but less robust.

The algorithm of chapter 4, which was published in [28], does not solve Wahba's

problem. The algorithm matches two images, the camera image and a database image that was generated using a coarse estimate of the attitude. Based on the needed translation and rotation to minimize the distance between corresponding stars, the algorithm calculates the attitude. Mathematically, one could say that AIM solves the 2D equivalent of Wahba's problem.

The advantage of this algorithm is that it is computationally less complex than the fastest state of the art algorithms. The disadvantage is the need for a coarse estimate and the negative impact of an error on this coarse estimate on the accuracy of the algorithm. In nominal operation of a star tracker, these disadvantages are shown to be insignificant.

In chapter 5, which was published in [30], the performance of this algorithm was increased further. The structure of the algorithm allows to reuse previously converted coordinates of the database stars. This leads to a significant reduction of the computational cost, for an algorithm that already had the lowest computational cost. A method to automatically detect and remove outliers, stars that have a large error in their centroiding position, greatly improves the robustness of the algorithm in an efficient way.

Applications

The novel algorithms provide improvements in robustness and computational cost, while offering accuracy that is comparable to that of the most accurate algorithms.

With the reduction in computational cost, a star tracker could provide the same accuracy as before by using less powerful and cheaper components. This makes the developed algorithms very useful for the rapidly growing field of small satellites, where computational power is less available. A star tracker with the developed algorithms could bring high pointing knowledge to these platforms.

The lower computational cost could also be used to track more stars or get the attitude estimates at a higher rate, which improves the accuracy. The higher accuracy can increase the scientific return of spacecraft, both by allowing for better pointing performance as by giving more accurate data for post-processing. This is especially beneficial for the higher-rang space missions, where pointing knowledge with high accuracy is very important.

The higher robustness allows to use the star tracker in more hostile environments (e.g. with high radiation, close to comets, etc.) and gives higher autonomy to the star tracker, reducing the ground segment cost. This opens up new perspective for missions that investigate comets or asteroids and missions that are in high radiation environments. The higher robustness is also useful in the

small satellite field. Small satellite missions have limited budgets for processing components, radiation shielding and redundancy.

7.1.2 CubeSat Attitude Determination and Control

The second contribution of this work is the development of a CubeSat Attitude Determination and Control System (ADCS). Compared to the main contribution, this work is much "broader". Rather than specifically aimed at innovation, it brings together the developments from several fields. Having said that, the CubeSat ADCS that is developed will have unprecedented pointing accuracy and will expand the range of missions that can be performed with a CubeSat.

The novelty of the KU Leuven ADCS lies in the addition of three reaction wheels and a star tracker to the set of sensors (gyroscope, photodiodes, magnetometer) and actuators (three magnetorquers) used in most existing CubeSat ADCS systems. The star tracker and three reaction wheels bring the pointing accuracy of the CubeSat from around 5 to 10 degrees to 1 degree and below. This pointing accuracy makes Earth observation missions possible.

The KU Leuven ADCS will be used in the SIMBA mission that measures the incoming and outgoing radiation of the Earth. The SIMBA mission is one of the first CubeSat missions to be supervised by ESA. A lot of effort is invested in the analysis of the performance, in the test campaign and in documenting the project. The old rule, saying that the pile of documentation for a satellite project is larger than the satellite itself, definitely holds for this project.

The development of the KU Leuven ADCS has an important educational value. More than 10 master thesis student and several other students have contributed to the project. They have gained hands-on experience with satellite development and were often motivated to continue in this field.

7.2 Future work

The field of (small) satellites is evolving rapidly and will most likely continue to do so in the coming years. The potential for interesting future work is enormous. The following section starts with suggestions for future work in the research field of this dissertation and ends with a more broad view.

7.2.1 Star Tracker Development

Algorithms

A first step towards further improvement of the star tracker algorithm cycle was given in [25]. In all current ADCS systems, the information of the star tracker and gyroscope is obtained separately and is fused in an estimator such as an Extended Kalman Filter. The method of [25] proposes to fuse the gyroscope and star tracker information earlier in the algorithm sequence of the star tracker, namely after the centroiding algorithm. This approach increases performance by adjusting the novel Kalman Filter parameters using certain available information. Using information such as the power in the star signal or the shape of the signal, the noise values in the filter can be adjusted to improve the star position estimate. Furthermore, the filter also estimates the uncertainty on the star positions. Knowledge of the uncertainty can be used to assign more importance to stars with lower position uncertainty in the cost function of the tracking algorithm. The Kalman Filter with these improvements was implemented and tested with simulated star data. Preliminary results show that the attitude estimation error is reduced significantly. This results in a more accurate attitude determination and control system which allows to perform more demanding missions. Future research can further improve and test this methodology.

Baffle

The ray tracing methodology to improve baffle performance by using new vane structures has shown a lot of potential in first tests. Further research will improve the ray tracing software and the performance of the obtained vane design, leading to baffles with higher stray light rejection. With the leading expertise in additive manufacturing available at KU Leuven, it should be possible to manufacture baffles that improve upon the state of the art in small satellite baffles. The market of small satellites is growing rapidly and the share of Earth observation missions (which have optical instruments that benefit from baffles) is increasing.

Build a Star Tracker

A project can be set up to build and market a star tracker. Particularly of interest is the market of small satellites. As mentioned in chapter 6, there are currently no highly accurate star trackers for the rapidly growing range of $<$10 kg satellites. These satellites require increasingly accurate attitude determination, which could be offered by a star tracker. A partnership with a company that has the facilities and expertise to build the star tracker could turn the developments of this work into an interesting star tracker product within a two year time frame.

7.2.2 CubeSat ADCS

Reaction Wheels

Next to the star tracker, the reaction wheels are the other major in-house development in the KU Leuven ADCS. The design of this component has largely been done by master thesis students. The results of vibration and thermal vacuum tests will be valuable to identify areas for improvement. The current reaction wheels could be made more compact by moving to a different motor design and reductions in friction could be obtained by adapting the rotor-housing design.

Other ADCS Parts

The KU Leuven ADCS uses the CubeSat design philosophy in the sense that it focusses on the development of some new parts (the star tracker and reaction wheels) and procures the other parts off-the-shelf. The KU Leuven ADCS currently uses a low-grade gyroscope and magnetometer. These sensors were selected in master thesis projects, where low cost and easy usage were preferred. A new analysis could lead to sensors with higher accuracy, which would benefit the ADCS performance.

Given the uncertainty of the size of certain components, some margin was taken in the mechanical design of the ADCS. Once the ADCS is fully integrated, these margins can be evaluated and the design could be made more compact. Currently the system fits within 0.8 unit (80mm height), which is convenient to mount within the CubeSat structure. If 15 mm could be gained in height, an extra CubeSat PC104 PCB could be added on the stack, which increases flexibility.

Test Campaign

An extensive test campaign is planned and needs to be performed in the near future. This test campaign includes performance as well as environmental (thermal vacuum, vibrations, etc.) tests. A successful test campaign increases the confidence in the system and gives valuable feedback to improve the design.

Launch and Follow-up

Once in orbit, the ADCS will control the attitude of the SIMBA CubeSat. The telemetry of the ADCS and the entire CubeSat will give very valuable information about the performance of the system. This information can be used to improve the design and can also be used to improve the ADCS attitude and orbit simulator and make it more realistic.

7.2.3 Small Satellite Research at KU Leuven

The lower cost and development time of CubeSats has made space research and satellite development accessible for universities. The field is high-tech and expanding rapidly. Space agencies are getting increasingly involved and industry is beginning to invest heavily. It is hard to imagine an environment which is better suited for engineering researchers. The great interest of master thesis students for the CubeSat project proposals shows the attractiveness of this field each year again.

At this point, KU Leuven has established a nice position within the, still young, small satellite market. The involvement in the world of small satellites (for KU Leuven specifically in the ADCS part of it) opens doors to future opportunities. The work on star trackers and an attitude determination and control system has given KU Leuven the opportunity to be part of a CubeSat consortium and has positioned it as a partner for future missions, responses to tenders, etc.

There is a solid base we can build on.

Continue ADCS Development

The logical step is to continue the development of the CubeSat ADCS. The future work mentioned in the paragraphs above discusses what the next steps are to finalise the current ADCS and how important parts of it can be improved. After a successful qualification of the ADCS in tests and on the SIMBA mission, it can be used in other CubeSats. This can be a source of income, as well as a way to set up partnerships with other institutes. The high involvement of Belgium in small satellite research (Von Karman Institute, Redu, RMIB, etc.) is a clear advantage.

The ADCS can be further expanded with a Sun sensor and Earth sensor. Both of these developments can be initialized in master thesis projects. Deployable or inflatable baffle structures are currently seen as complex. With the expertise in micro-mechanics and light structures available at KU Leuven, a good design can surely be made.

The in-house developed components: the star tracker, reaction wheels, baffle and Z-magnetorquer can be sold as separate components to other satellite developers. Other universities are using this approach with success. The components developed at KU Leuven improve in many respects upon the state of the art, so there is definitely potential here.

Developments linked to ADCS

Stepping away from the ADCS itself, a lot of interesting opportunities that are still linked to the attitude and orbit of satellites become available.

A research path that has already been initiated in a master thesis project [11] is the development of an accurate pointing platform for the payload. A piezo-actuated stage translates the lens of an optical payload to compensate for changes in attitude that cannot be compensated by the ADCS. This leads to higher pointing stability for the payload, which can then deliver higher quality images. This further opens up the potential for astronomical missions for CubeSats. The good collaboration with the Institute for Astronomy could be expanded further in the development of such a CubeSat.

A very interesting future prospect for small satellites is the docking with other satellites. This way, large structure in space can be built using smaller satellites that can be more flexibly launched for a lower cost. Another potential application is to build modular satellites. The life time of satellites is often limited by the life time of the reaction wheels. A modular satellite could discard its reaction wheel module and replace it with a new one that docks to the satellite. Large satellites that now need to end their mission because they run out of fuel could be refueled using small satellites that dock to it. At the moment this sounds quite futuristic, but now is the time to start developing the first techniques to do this.

Experience has already been built up with developing an attitude and orbit simulator and with attitude determination and control. The techniques that are necessary to rendez-vous and dock with a satellite could be developed, starting from this expertise. Setting up a coherent simulation environment for rendez-vous and docking or developing new algorithms is challenging and valuable research. Furthermore, it builds up expertise and knowledge at KU Leuven. At the University of Liège, a lot of work has been done on rendez-vous and orbit control, this could lead to an interesting collaboration.
It would also be interesting to think about the docking connection. To increase flexibility, a standard docking connection would be very beneficial. Analyzing the requirements and devising such a standard is incredibly interesting. The CubeSat standard was developed by a university (Cal Poly), why would KU Leuven not be able to develop a docking connector standard?
This work will give KU Leuven a head start when future tenders or industry interest regarding CubeSat docking comes up.

Overall Small Satellite developments

A satellite is generally only useful when it can communicate with a ground station. Most CubeSat teams only use one ground station, which leads to limited ground station visibility. The SIMBA CubeSat will be able to communicate with the ground station for an estimated 50 minutes per day [33].

Having more ground stations on different locations increases ground station visibility and the data budget of a satellite. It would be interesting for KU Leuven to build a ground station, potentially with the help of students. A ground station could be built in Leuven or in La Palma, where the Institute for Astronomy has its Mercator telescope. This location is interesting because there is a lower presence of ground stations in the vicinity. A ground station for CubeSats is extra effective near the poles. Because CubeSats often have polar orbits, they spend a higher amount of time over the poles. Given that Belspo is both responsible for the Belgian space policy as for the Prinses Elisabeth base on Antarctica, it could be interesting to work together with them on this. A system and business plan could be set up to make the ground stations available for other satellite operators. The value of a ground station in negotiations for participation in a CubeSat consortium should also not be underestimated.

KU Leuven could be the prime investigator for a CubeSat of its own. The Institute for Astronomy could fly an astronomical mission, the Department of Biology could fly a biological payload, the Department of Geography could benefit from Earth observation, etc. Experience has already been built up with CubeSat development and there are contacts within the field to make such a project successful. A satellite of its own would be a nice addition to the history of KU Leuven. Maybe we should change the slogan:

“Discover yourself, start with the world, but don’t stop there.”

Chapter 8

Gaussian Grid A and B parameters

In this appendix, the expressions to calculate the values for $A_{i,j}$ and $B_{i,j}$ are given for the 3×3, 5×5, and 7×7 pixel window sizes. The expressions are a function of the weights w_i used in the cost function. These weights are assigned as follows:

Table 8.1: Weights grid.

value	V_{x_c-3,y_c}	V_{x_c-2,y_c}	V_{x_c-1,y_c}	V_{x_c,y_c}	V_{x_c+1,y_c}	V_{x_c+2,y_c}	V_{x_c+3,y_c}
w	w_{-3}	w_{-2}	w_{-1}	w_0	w_1	w_2	w_3

3×3 pixel window

The values for $A_{i,j}$ and $B_{i,j}$ for a 3×3 pixel window are calculated as follows:

$$B_0 = 0$$

$$B_1 = 1$$

$$B_{-1} = -1$$

$$A_0 = 4$$

$$A_1 = -2$$

$$A_{-1} = -2$$

$$C_0 = 2$$

$$C_1 = -1$$

$$C_{-1} = -1$$

$$D = 6$$

5×5 pixel window

The values for $A_{i,j}$ and $B_{i,j}$ for a 5×5 pixel window are calculated as follows:

$$
\begin{aligned}
A_0 = {} & 2(w_0w_1w_2 + w_0w_{-1}w_{-2}) - 4w_0w_1w_{-1} \\
& - 18(w_0w_1w_{-2} + w_0w_{-1}w_2) - 64w_0w_2w_{-2} \\
A_1 = {} & 2w_0w_{-1}w_1 + 6w_1w_{-1}w_{-2} - 4w_0w_1w_2 \\
& + 12w_0w_1w_{-2} - 18w_1w_2w_{-1} - 48w_1w_2w_{-2} \\
A_2 = {} & 2w_0w_1w_2 + 6w_0w_2w_{-1} + 12(w_1w_2w_{-1} + w_2w_{-1}w_{-2}) \\
& + 32w_0w_2w_{-2} + 36w_1w_2w_{-2} \\
A_{-1} = {} & 2w_0w_1w_{-1} + 6w_1w_2w_{-1} - 4w_0w_{-1}w_{-2} + 12w_0w_2w_{-1} - 18w_1w_{-1}w_{-2} \\
& - 48w_2w_{-1}w_{-2} \\
A_{-2} = {} & 2w_0w_{-1}w_{-2} + 6w_0w_{-2}w_1 + 12(w_1w_2w_{-2} + w_1w_{-1}w_{-2}) + 32w_0w_2w_{-2} \\
& + 36w_2w_{-1}w_{-2} \\
B_0 = {} & 3(w_0w_1w_2 - w_0w_{-1}w_{-2}) + 9(w_0w_1w_{-2} - w_0w_{-1}w_2) \\
B_1 = {} & -w_0w_1w_{-1} - 4w_0w_1w_2 - 9(w_1w_2w_{-1} + w_1w_{-2}w_{-1}) - 12w_0w_1w_{-2} \\
B_2 = {} & w_0w_1w_2 - 3w_0w_{-1}w_2 - 18(w_1w_{-2}w_2 + w_{-1}w_{-2}w_2) - 32w_0w_2w_{-2} \\
B_{-1} = {} & w_0w_1w_{-1} + 4w_0w_{-1}w_{-2} + 9(w_1w_2w_{-1} + w_1w_{-1}w_{-2}) + 12w_0w_2w_{-1} \\
B_{-2} = {} & -w_0w_{-1}w_{-2} + 3w_0w_1w_{-2} + 18(w_1w_2w_{-2} + w_2w_{-1}w_{-2}) + 32w_0w_2w_{-2} \\
C_0 = {} & -32w_2w_{-2} - 9(w_1w_{-2} + w_2w_{-1}) + w_{-1}w_{-2} + w_1w_2 \\
& - 2w_1w_{-1} \\
C_1 = {} & -2w_0w_2 + w_0w_{-1} + 6w_0w_{-2} - 9w_2w_{-1} - 24w_2w_{-2} + 3w_{-1}w_{-2} \\
C_2 = {} & w_0w_1 + 3w_0w_{-1} + 16w_0w_{-2} + 18w_1w_{-2} + 6(w_1w_{-1} + w_{-1}w_{-2}) \\
C_{-1} = {} & w_0w_1 + 6w_0w_2 - 2w_0w_{-2} + 3w_1w_2 - 9w_1w_{-2} - 24w_2w_{-2} \\
C_{-2} = {} & 18w_2w_{-1} + 6(w_1w_{-1} + w_1w_2) + 16w_0w_2 + w_0w_{-1} + 3w_0w_1 \\
D = {} & -128w_0w_2w_{-2} - 72w_2(w_1w_{-2} + w_{-1}w_{-2}) \\
& - 18(w_0w_1w_{-2} + w_2w_{-1}w_0 + w_1w_{-2}w_{-1} + w_1w_2w_{-1}) \\
& - 2w_0(w_1w_2 + w_1w_{-1} + w_{-1}w_{-2})
\end{aligned}
$$

7×7 pixel window

$$
\begin{aligned}
A_{-3} =& 12w_0w_1w_{-3} + 60w_0w_2w_{-3} + 162w_0w_3w_{-3} + 6w_0w_{-1}w_{-3} \\
&+ 12w_0w_{-2}w_{-3} + 20w_1w_2w_{-3} + 96w_1w_3w_{-3} + 32w_1w_{-1}w_{-3} \\
&+ 30w_2w_3w_{-3} + 90w_2w_{-1}w_{-3} + 80w_2w_{-2}w_{-3} + 192w_3w_{-1}w_{-3} \\
&+ 150w_3w_{-2}w_{-3} + 2w_{-1}w_{-2}w_{-3} + 36w_1w_{-2}w_{-3} \\
A_{-2} =& 2w_0w_{-1}w_{-2} - 18w_0w_{-2}w_{-3} + 12w_1w_2w_{-2} + 60w_1w_3w_{-2} \\
&+ 12w_1w_{-1}w_{-2} - 48w_1w_{-2}w_{-3} + 20w_2w_3w_{-2} + 36w_2w_{-1}w_{-2} \\
&- 100w_2w_{-2}w_{-3} + 80w_3w_{-1}w_{-2} - 180w_3w_{-2}w_{-3} - 4w_{-1}w_{-2}w_{-3} \\
&+ 6w_0w_1w_{-2} + 32w_0w_2w_{-2} + 90w_0w_3w_{-2} \\
A_{-1} =& 2w_0w_1w_{-1} + 12w_0w_2w_{-1} + 36w_0w_3w_{-1} - 4w_0w_{-1}w_{-2} \\
&+ 6w_1w_2w_{-1} + 32w_1w_3w_{-1} - 18w_1w_{-1}w_{-2} - 64w_1w_{-1}w_{-3} \\
&- 18w_0w_{-1}w_{-3} + 48w_2w_{-1}w_{-2} - 150w_2w_{-1}w_{-3} - 100w_3w_{-1}w_{-2} \\
&- 288w_3w_{-1}w_{-3} + 2w_{-1}w_{-2}w_{-3} + 12w_2w_3w_{-1} \\
A_0 =& 2w_0w_1w_2 + 12w_0w_1w_3 - 4w_0w_1w_{-1} - 18w_0w_1w_{-2} - 48w_0w_1w_{-3} + \\
&6w_0w_2w_3 - 18w_0w_2w_{-1} - 64w_0w_2w_{-2} - 150w_0w_2w_{-3} - 48w_0w_3w_{-1} - \\
&150w_0w_3w_{-2} - 324w_0w_3w_{-3} + 2w_0w_{-1}w_{-2} + 12w_0w_{-1}w_{-3} + 6w_0w_{-2}w_{-3} \\
A_1 =& 36w_0w_1w_{-3} + 2w_1w_2w_3 - 18w_1w_2w_{-1} - 48w_1w_2w_{-2} - 100w_1w_2w_{-3} - \\
&64w_1w_3w_{-1} - 150w_1w_3w_{-2} - 288w_1w_3w_{-3} + 6w_1w_{-1}w_{-2} + 32w_1w_{-1}w_{-3} + \\
&12w_1w_{-2}w_{-3} - 4w_0w_1w_2 - 18w_0w_1w_3 + 2w_0w_1w_{-1} + 12w_0w_1w_{-2} \\
A_2 =& 2w_0w_1w_2 - 18w_0w_2w_3 + 6w_0w_2w_{-1} + 32w_0w_2w_{-2} + 90w_0w_2w_{-3} - \\
&4w_1w_2w_3 + 12w_1w_2w_{-1} + 36w_1w_2w_{-2} + 80w_1w_2w_{-3} - 48w_2w_3w_{-1} - \\
&100w_2w_3w_{-2} - 180w_2w_3w_{-3} + 12w_2w_{-1}w_{-2} + 60w_2w_{-1}w_{-3} + 20w_2w_{-2}w_{-3} \\
A_3 =& 6w_0w_1w_3 + 12w_0w_2w_3 + 12w_0w_3w_{-1} + 60w_0w_3w_{-2} + 162w_0w_3w_{-3} + \\
&2w_1w_2w_3 + 32w_1w_3w_{-1} + 90w_1w_3w_{-2} + 192w_1w_3w_{-3} + 36w_2w_3w_{-1} + \\
&80w_2w_3w_{-2} + 150w_2w_3w_{-3} + 20w_3w_{-1}w_{-2} + 96w_3w_{-1}w_{-3} + 30w_3w_{-2}w_{-3}
\end{aligned}
$$

$$
\begin{aligned}
B_{-3} =& 6w_0w_1w_{-3} + 60w_0w_2w_{-3} + 243w_0w_3w_{-3} - 3w_0w_{-1}w_{-3} - 12w_0w_{-2}w_{-3} \\
&+ 30w_1w_2w_{-3} + 192w_1w_3w_{-3} - 18w_1w_{-2}w_{-3} + 75w_2w_3w_{-3} \\
&+ 45w_2w_{-1}w_{-3} + 192w_3w_{-1}w_{-3} + 75w_3w_{-2}w_{-3} - 3w_{-1}w_{-2}w_{-3} \\
B_{-2} =& 3w_0w_1w_{-2} + 32w_0w_2w_{-2} + 135w_0w_3w_{-2} - w_0w_{-1}w_{-2} + 27w_0w_{-2}w_{-3} + \\
&18w_1w_2w_{-2} + 120w_1w_3w_{-2} + 48w_1w_{-2}w_{-3} + 50w_2w_3w_{-2} \\
&+ 18w_2w_{-1}w_{-2} + 50w_2w_{-2}w_{-3} + 80w_3w_{-1}w_{-2} + 8w_{-1}w_{-2}w_{-3} \\
B_{-1} =& w_0w_1w_{-1} + 12w_0w_2w_{-1} + 54w_0w_3w_{-1} + 4w_0w_{-1}w_{-2} + 27w_0w_{-1}w_{-3} + \\
&9w_1w_2w_{-1} + 64w_1w_3w_{-1} + 9w_1w_{-1}w_{-2} + 64w_1w_{-1}w_{-3} + 30w_2w_3w_{-1} + \\
&75w_2w_{-1}w_{-3} - 50w_3w_{-1}w_{-2} - 5w_{-1}w_{-2}w_{-3} \\
B_0 =& -15w_0w_{-2}w_{-3} + 3w_0w_1w_2 + 24w_0w_1w_3 + 9w_0w_1w_{-2} + 48w_0w_1w_{-3} + \\
&15w_0w_2w_3 - 9w_0w_2w_{-1} + 75w_0w_2w_{-3} - 48w_0w_3w_{-1} - 75w_0w_3w_{-2} - \\
&3w_0w_{-1}w_{-2} - 24w_0w_{-1}w_{-3} \\
B_1 =& -75w_1w_3w_{-2} - 9w_1w_{-1}w_{-2} - 64w_1w_{-1}w_{-3} - 30w_1w_{-2}w_{-3} \\
&- 4w_0w_1w_2 - 27w_0w_1w_3 - w_0w_1w_{-1} - 12w_0w_1w_{-2} - 54w_0w_1w_{-3} \\
&+ 5w_1w_2w_3 - 9w_1w_2w_{-1} + 50w_1w_2w_{-3} - 64w_1w_3w_{-1} \\
B_2 =& w_0w_1w_2 - 27w_0w_2w_3 - 3w_0w_2w_{-1} - 32w_0w_2w_{-2} - 135w_0w_2w_{-3} - \\
&8w_1w_2w_3 - 18w_1w_2w_{-2} - 80w_1w_2w_{-3} - 48w_2w_3w_{-1} - 50w_2w_3w_{-2} - \\
&18w_2w_{-1}w_{-2} - 120w_2w_{-1}w_{-3} - 50w_2w_{-2}w_{-3} \\
B_3 =& 3w_0w_1w_3 + 12w_0w_2w_3 - 6w_0w_3w_{-1} - 60w_0w_3w_{-2} - 243w_0w_3w_{-3} + \\
&3w_1w_2w_3 - 45w_1w_3w_{-2} - 192w_1w_3w_{-3} + 18w_2w_3w_{-1} - 75w_2w_3w_{-3} - \\
&30w_3w_{-1}w_{-2} - 192w_3w_{-1}w_{-3} - 75w_3w_{-2}w_{-3}
\end{aligned}
$$

Chapter 9

Disturbance Torques

In this appendix, the disturbance torques that act upon the SIMBA CubeSat are calculated. Worst-case calculations are made. These values were used in the performance simulations.

Disturbance torques can be divided in two groups [53]:

- Cyclic: varying in a sinusoidal manner during an orbit and integrate to zero over an integer number of cycles.
- Secular: not periodic or averaging out over an orbit and accumulating with time.

9.1 Solar Pressure

The solar pressure torque T_{sp} is cyclic for earth-oriented spacecraft since the exposed surface varies. A constant exposed surface for solar-oriented spacecraft results in a secular disturbance. Because SIMBA will be earth-oriented for the majority of the mission duration, T_{sp} is considered to be cyclic.
T_{sp} is mostly influenced by the spacecraft geometry, surface properties and center of gravity location. The solar pressure is calculated as [134]:

$$T_{sp} = F.\Delta_{cp,cg} \tag{9.1}$$

Where:

$$F = \frac{F_s}{c} A_s \cos(i)(1+q) \tag{9.2}$$

The values used are given in Table 9.1.

Table 9.1: The values used in the calculation of the solar pressure disturbance torque.

Parameter	Description	Value	Unit
F_s	Solar constant	1367	$\frac{W}{m^2}$
c	Speed of light	3.10^8	$\frac{m}{s}$
A_s	Maximal Surface	0.05	m^2
$\Delta_{cp,cg}$	Distance between center of gravity and center of pressure	0.03	m
q	Reflection Factor	0.5	-
i	Incident angle of the Sun	0	deg

9.2 Gravity Gradient

The gravity gradient torque T_{gg} is secular for earth-oriented spacecraft where the angle between the nadir vector and body axis remains constant. A constant external torque is generated since the lower extremities compared to the higher are exposed to higher gravity forces. They become cyclic for inertial pointed spacecraft where the angle is varying. Because SIMBA will be earth-oriented for the majority of the mission duration, T_{gg} is considered to be secular.
Besides inertia, T_{gg} is primarily influenced by the orbit altitude and calculated as [134]:

$$T_{gg} = \frac{3\mu}{2(R+h)^3} \sin(2\theta_{or}) |I_z - I_y| \tag{9.3}$$

The values used are given in Table 9.2.

Table 9.2: The values used in the calculation of the gravity gradient disturbance torque.

Parameter	Description	Value	Unit
μ	Standard gravitational parameter of Earth	398600	$\frac{km^3}{s^2}$
I_x	Moment of Inertia around the x-axis	0.005	kgm^2
I_y	Moment of Inertia around the y-axis	0.025	kgm^2
I_z	Moment of Inertia around the z-axis	0.025	kgm^2
θ_{or}	Orientation of the CubeSat	45	deg
R	Radius of the Earth	6378.10^3	m

9.3 Magnetic Field

The magnetic field T_{mf} is a cyclic disturbance created by the residual spacecraft magnetic dipole D (from magnetic material and current loops) which tends to align with the earth magnetic field. T_{mf} has a period of a complete orbit for earth pointing spacecraft and a half when sun pointing.

The magnitude of D, orbit altitude and inclination are the main parameters influencing T_{mf} expressed by [134].

$$T_{mf} = DB \tag{9.4}$$

Where

$$B = \frac{2M}{(R+h)^3} \tag{9.5}$$

The difficulty in calculating the T_{mf} is determining D, which is generally obtained by testing. Reference [134] states that a typical values for small, uncompensated spacecraft are 1 Am^2. A small calculation for a 3U cubesat of 3W, consisting of 19 PCB layers operating at 3.3 - 9V learns that there has to be 127 - 347 complete current loops/PCB (with maximum i and A) to reach this value. A mean of ± 10 loops/PCB or D = 0.05Am2 seems to be a more acceptable value to incorporate the current loops and magnetic material. A similar value is found when Dipole values used in [49, 50, 45, 5] are averaged, taking into account the number of units.
The values used are given in Table 9.3.

Table 9.3: The values used in the calculation of the magnetic field disturbance torque.

Parameter	Description	Value	Unit
D	Dipole of the satellite	0.05	Am^2
M	Magnetic dipole moment of the Earth	$7.96.10^{15}$	Tm^3
R	Radius of the Earth	6378.10^3	m

9.4 Aerodynamic Drag

The final disturbance torque is the torque created by the aerodynamic drag, T_{ad}. T_{ad} is secular.

$$T_{ad} = F.\Delta_{cp,cg} \tag{9.6}$$

Where

$$F = \frac{\rho C_d A_f V^2}{2} \tag{9.7}$$

Orbit altitude and center of gravity location are dominating T_{ad}, given by [134]. ρ is rapidly decreasing with altitude and the most difficult to determine since it is varying with solar activity, atmospheric conditions, seasons, latitude, etc. A worst case value for the selected altitudes is taken from [134]. The values used are given in Table 9.4.

Table 9.4: The values used in the calculation of the aerodynamic drag disturbance torque.

Parameter	Description	Value	Unit
C_d	Drag Coefficient	2.2	-
ρ	Atmospheric density at 600 km	$1.47.10^{-13}$	$\frac{kg}{m^3}$
A_f	Frontal surface	0.01	m^2
$\Delta_{cp,cg}$	Distance between center of gravity and center of pressure	0.03	m
V	Velocity of the satellite	7.558	$\frac{km}{s}$

9.5 Total Disturbance Torque

The worst case values of these disturbance torques for the SIMBA mission are given in table 9.5.

Table 9.5: The values used in the calculation of the aerodynamic drag disturbance torque.

Torque	Cause	Value	Unit
T_{sp}	Solar Pressure	$1.03.10^{-8}$	Nm
T_{gg}	Gravity Gradient	$3.52.10^{-8}$	Nm
T_{mf}	Magnetic Field	$2.34.10^{-6}$	Nm
T_{ad}	Atmospheric Drag	$9.14.10^{-9}$	Nm
T_{sec}	Total Secular Torque	$4.43.10^{-8}$	Nm
T_{cyc}	Total Cyclic Torque	$2.35.10^{-6}$	Nm
T_{tot}	Total Torque	$2.40.10^{-6}$	Nm

A graph showing simulated disturbance torques during 4 orbits is given in figure 9.1

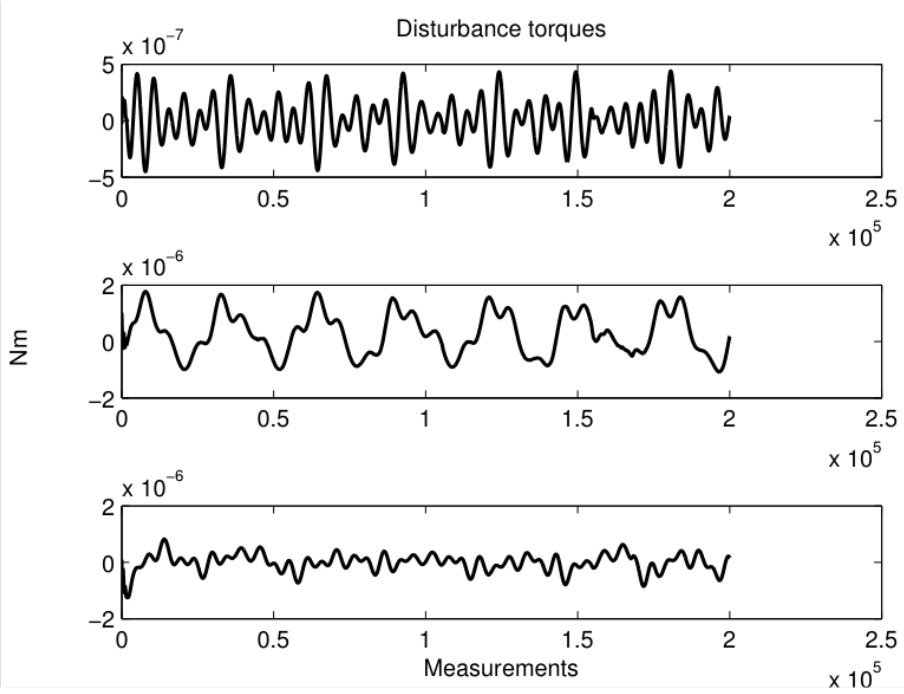


Figure 9.1: Typical disturbance torques for the SIMBA CubeSat generated during the simulation over 4 orbits.

Bibliography

[1] D. P. A. Landi, M. Buerni. Star trackers for accurate pointing and attitude determination - in-flight results of high-accuracy star trackers of iso and sax - new development towards navigation autonomy. In *Proceedings of the 3rd ESA International Conference held 26-29 November*, page 621, ESTEC, Noordwijk, November 1997.

[2] C. Al-Ekabi, B. Baranes, P. Hulsroj, and A. Lahcen. *Yearbook on Space Policy 2011/2012: Space in Times of Financial Crisis*. Springer Wien, New York NY, 2014.

[3] P. Alvelda and A. M. San Martin. Neural network star pattern recognition for spacecraft attitude determination and control. In *Advances in Neural Information Processing Systems*, pages 314–322, 1989.

[4] N. Ames. Bigelow spacecraft carries nasa 'genebox' to orbit. `http://www.nasa.gov/centers/ames/multimedia/images/2006/genebox_prt.htm`, [retrieved 25 august 2015].

[5] R. Amini, F. Patrick, K. Kjeldgaard, J. Kjaer, Y. Liu, and M. Villekjaer. Attitude determination system for aausat-ii. Technical report, Institute of Electronic System AALBORG University. Report of Group: 04gr830b, 2004.

[6] D. Anderson. *Autonomous Star Sensing and Pattern Recognition for Spacecraft Attitude Determination*. PhD thesis, A&M University Texas, 1991.

[7] L. Auer and W. V. Altena. Digital image centering ii. *The Astronomical Journal*, 83(5), 1978.

[8] R. L. Baer. A model for dark current characterization and simulation. In *2006 Electronic Imaging Conference: SPIE Vol. 6068*, 2006.

[9] BallAerospace. Ct-602 star tracker. `http://www.ballaerospace.com/file/media/D0540_CT-602.pdf`, 2015.

[10] Berlin Space Tech. iadcs-100 intelligent attitude control for cubesats. Technical report, Berlin Space Tech, 2015.

[11] M. Bert. Design and implementation of a high-precision pointing platform for a cubesat payload. Master's thesis, KU Leuven, 2015.

[12] Blue Canyon Technologies. Xact - high-performance attitude determination for cubesats. Technical report, Blue Canyon Technologies, 2015.

[13] Blue Canyon Technologies. Xact lite - attitude control for cubesats. Technical report, Blue Canyon Technologies, 2015.

[14] E. Buchen. Small satellite market observations. In *Proceedings of the 29th Annual AIAA/USU Conference on Small Satellites*, 2015.

[15] E. Buchen and D. DePasquale. 2014 nano/microsatellite market assessment. Technical report, SpaceWorks Enterprises, inc., 2014.

[16] Y. Cheng and M. D. Shuster. Robustness and accuracy of the quest algorithm. In *AAS/AIAA 17th Space Flight Mechanics Meeting*, pages 46–61, 2007.

[17] Y. Cheng and M. D. Shuster. Speed testing of attitude estimation algorithms. submitted to The Journal of the Astronautical Sciences, 2008.

[18] Chris Adolphus Esionwu Jr. Cubesat market analysis. Technical report, Kingston University London, 2013.

[19] L. Chunyan, L. Ke, Z. Longyun, J. Shengzhen, and Z. Jifeng. Star pattern recognition method based on neural network. *Chinese Science Bulletin*, 48(18):1927–1930, 2003.

[20] ClydeSpace. Clyde space products. `http://www.clyde-space.com/products`, [retrieved 28 July 2015].

[21] N. R. Council. *Reducing the costs of space science research missions*. National Academy press, 3 edition, 1997. proceedings of a workshop, joint committee on Technology for Space Science and Applications of the Aeronautics and Space Engineering Board and the Space Studies Board.

[22] J. Crassidis and J. Junkins. *Optimal Estimation of Dynamic Systems: Second Edition*. Chapman & Hall CRC Press, Boca Raton, FL, 2012.

[23] R. De Nutte. Design and development of a prototype star tracker using advanced attitude estimation algorithms. Master's thesis, KU Leuven, Celestijnenlaan 300B, 3001 Heverlee, Belgium, 2014.

[24] T. Delabie. A highly efficient attitude estimation algorithm for star trackers based on optimal image matching. In *AIAA Guidance, Navigation and Control Conference, Minneapolis, Minnesota*, 2012.

[25] T. Delabie. Star position estimation improvements for accurate star tracker attitude estimation. In *Scitech 2015*, 2015.

[26] T. Delabie, T. Durt, and J. Vandersteen. A highly robust lost in space algorithm based on the shortest distance transform. In *AIAA Guidance, Navigation, and Control Conference*, 2011.

[27] T. Delabie, T. Durt, and J. Vandersteen. Highly robust lost-in-space algorithm based on the shortest distance transform. *Journal of Guidance, Control and Dynamics*, 36(2):476–484, 2013.

[28] T. Delabie, J. D. Schutter, and B. Vandenbussche. Highly efficient attitude estimation algorithm for star trackers using optimal image matching. *Journal of Guidance, Control and Dynamics*, 32(6):1672–1680, 2013.

[29] T. Delabie, J. D. Schutter, and B. Vandenbussche. An accurate and efficient gaussian fit centroiding algorithm for star trackers. *The Journal of the Astronautical Sciences*, 2015.

[30] T. Delabie, J. D. Schutter, and B. Vandenbussche. Robustness and efficiency improvements for star tracker attitude estimation. *Journal of Guidance, Control and Dynamics*, 38(11):2108–2121, 2015.

[31] L. Dell'Elce and G. Kerschen. Optimal propellantless rendez-vous using differential drag. *Acta Astronautica*, 1:112–123, 2015.

[32] D. DePasquale and J. Bradford. Nano/microsatellite market assessment. Technical report, SpaceWorks Enterprises, inc., February 2013.

[33] S. Dewitte and T. Delabie. Mission analysis report. Technical report, RMIB, 2015.

[34] S. Dewitte and T. Delabie. System requirements document of the simba cubesat. Technical report, RMIB, 2015.

[35] A. Duflos. Business case analysis of a multi-functional in-flight experimental system. Technical report, Euroconsult & Astrium ST, 2014.

[36] ESA. The hipparcos and tycho catalogues. Technical report, ESA SP-1200, 1997.

[37] ESA. Rosetta status report: Close flyby navigation issues. `http://blogs.esa.int/rosetta/2015/04/01/rosetta-status-report-close-flyby-navigation-issues/`, 2015.

[38] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno. 2D euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys*, 40(1):44, February feb 2008.

[39] Faulhaber. *Brushless Flat DC-Micromotors 0,6 mNm*, 2015.

[40] T. G. Feeman. *Portraits of the Earth: A mathematician looks at maps*. The American Mathematical Society, 2002. ISBN: 0-8218-3255-7.

[41] D. Felikson, J. A. Hahmall, M. F. Vess, and M. Ekinci. On-orbit solar dynamics observatory (sdo) star tracker warm pixel analysis. In *AIAA Guidance, Navigation and Control Conference, Portland, Oregon*, 2011.

[42] H. Feuchtgruber. Herschel str-a ccd sub-pixel structure. Technical report, Max Planck Institute for Extraterrestrial Physics PICC-ME-TN-041, Garching, Germany, 2012.

[43] C. Finlay, S. Maus, C. Beggan, M. Hamoudi, F. Lowers, N. Olsen, and E. Thebault. Evaluation of candidate geomagnetic field models for igrf-11. *Earth, planets and space*, 62(10):787–804, 2010.

[44] T. Flatley, W. Morgenstern, A. Reth, and F. Bauer. A b-dot acquisition controller for the radarsat spacecraft. In *NASA Conference Publication*, pages 79–90. NASA, 1997.

[45] A. Foletti and P. Kaewkerd. Swisscube phase a adcs report. Technical report, EFPL Lausanne, Switzerland, 2006.

[46] A. Fontana. Business case analysis of a multi-functional in-flight experimental system. Technical report, Matt MacDonald, ELV and Avio, 2014.

[47] L. S. Forczyk. Swiss space systems (s3). In *The Space Congress Proceedings*, 2015.

[48] C. Fosu, G. Hein, and B. Eissfeller. Determination of centroid of ccd star images. University FAF Munich Germany, Commission III, WG III/8, 2004.

[49] A. Garza. Reaction wheels for picosatellites. Master's thesis, Lulea University of Technology, Sweden, 2009.

[50] J. Giesselmann. Development of an active magnetic attitude determination and control system for picosatellites on highly inclined circular low earth orbits. Master's thesis, RMIT University, Australia, 2006.

[51] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, MD, 1983.

[52] GOMspace. Gomspace cubesat products. `http://gomspace.com/index.php?p=products`, [retrieved 25 August 2015].

[53] J. Goyvaerts. Design and implementation of a low-cost attitude determination and control system for cubesats. Master's thesis, KU Leuven, Celestijnenlaan 300B, 3001 Heverlee, Belgium, 2011.

[54] GRW. High-precision ball bearings product catalogue. `http://www.grw.de/en/services/downloads.html`, 2015.

[55] Z. Guangjun, X. Wei, and J. Jiang. Full-sky autonomous star identification based on radial and cyclic features of star pattern. *Image Vision Comput.*, 26:891–897, 2008.

[56] B. R. Hancock, R. C. Stirbl, T. J. Cunningham, B. Pain, C. J. Wrigley, and P. G. Ringold. Cmos active pixel sensor specific performance effects on star tracker/imager position accuracy. In *Symposium on Integrated Optics*, pages 43–53. International Society for Optics and Photonics, 2001.

[57] R. Hardin, N. Sloane, and W. Smith. Tables of spherical codes with icosahedral symmetry. `http://www.research.att.com/~njas/icosahedral.codes/`, [retrieved 15 march 2010].

[58] H. Helvajian and S. W. Janson. *Small Satellites: Past, Present and Future*. The Aerospace Press, El Segundo, California, 2008.

[59] J. Hong and J. A. Dickerson. Neural-network-based autonomous star identification algorithm. *Journal of Guidance, Control and Dynamics*, 23(4):728–735, 2000.

[60] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[61] H. I. inc. *3-Axis Digital Compass IC HMC5883L*, 2011.

[62] I. Inc. *ITG-3200 Product Specification Revision 1.4*, 2010.

[63] S. Inc. *SLCD-61N8 Solderable Planar Photodiode*, 2015.

[64] P. Incorporated. Pl-1 pumpkin price list. Technical report, Pumpkin Incorporated, 2015.

[65] Innovative Solutions in Space. Cubesatshop. `http://www.cubesatshop.com`, [retrieved 28 July 2015].

[66] J. R. Janesick. *Scientific Charge-Coupled Devices*. SPIE - The international Society for Optical Engineering, Bellingham, Washington USA, 2001.

[67] I. Klotz. Space junk reaching "tipping point," report warns. http://www.reuters.com/article/2011/09/01/us-space-debris-idUSTRE7805VY20110901, last checked on 2012-04-27, 2011.

[68] I. Lafaille. Design and implementation of an attitude estimation and control algorithm for cubesats. Master's thesis, KU Leuven, Celestijnenlaan 300B, 3001 Heverlee, Belgium, 2014.

[69] W. Lan, R. Munakata, R. Nugent, and D. Pignatelli. Poly picosatellite orbital deployer mk. iii rev. e user guide. Technical report, Cal Poly SLO, 2014.

[70] G. A. Landis, S. R. Oleson, M. L. McGuire, L. M. Burke, M. C. Martini, J. E. Fittje, and T. W. Packard. A cubesat asteroid mission: Propulsion trade-offs. In *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, 2014.

[71] M. F. Landis. Attitude determination using vector observations and the singular value decomposition. *Journal of the Astronautical Sciences*, 36(3):245–258, July-Sept. 1988.

[72] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[73] C. Liebe. Pattern recognition of star constellations for spacecraft applications. *Aerospace and Electronic Systems Magazine, IEEE*, 7(6):34–41, 1992.

[74] C. C. Liebe. Star trackers for attitude determination. *Aerospace and Electronic Systems Magazine, IEEE*, 10(6):10–16, 1995.

[75] C. C. Liebe. Accuracy performance of star trackers-a tutorial. *IEEE Transactions on Aerospace and Electronic Systems*, pages 587–599, 2002.

[76] A. Limited. *Cortex-M4 Technical Reference Manual*, 2010.

[77] L. G. M. Three-axis attitude determination. In *Spacecraft Attitude Determination and Control*, pages 420–428. Springer Scientific + Business Media, Berlin and New York, 1978.

[78] F. L. Markley. Attitude determination using vector observations: a fast optimal matrix algorithm. *The Journal of the Astronautical Sciences*, 36(3):261–280, 1993.

[79] F. L. Markley and D. Mortari. How to estimate attitude from vector observations. In *AAS/AIAA Astrodynamics Specialist Conf.*, number 427 in 99, 1999.

[80] Maryland Aerospace Inc. Mai-100 miniature 3-axis adacs. Technical report, Maryland Aerospace Inc., 2015.

[81] Maryland Aerospace Inc. Mai-200 miniature 3-axis adacs. Technical report, Maryland Aerospace Inc., 2015.

[82] Maryland Aerospace Inc. Mai-400 1/2u cubesat adcs. Technical report, Maryland Aerospace Inc., 2015.

[83] C. R. Maurer Jr, R. Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(2):265–270, 2003.

[84] I. S. McLean. *Electronic Imaging in Astronomy: detectors and Instrumentation (Second Edition)*. Praxis, Chichester, UK, 2008.

[85] J. H. Meeus. *Astronomical algorithms*. Willmann-Bell, Incorporated, 1991.

[86] D. Mortari. Esoq: A closed-form solution to the wahba problem. *The Journal of the Astronautical Sciences*, 45(2):195–204, 1997.

[87] D. Mortari. Esoq2: Single-point algorithm for fast optimal attitude determination. *Advances in the Astronautical Sciences*, 97(2):803–816, 1997.

[88] D. Mortari. Second estimator of the optimal quaternion. *Journal of Guidance, Control, and Dymanics*, 23(5):885–887, 2000.

[89] D. Mortari, M. A. Samaan, C. Bruccoleri, and J. L. Junkins. The pyramid star identification technique. *Navigation*, 51(3):171–183, 2004.

[90] J. Mueller, R. Hofer, and J. Ziemer. Survey of propulsion technologies applicable to cubesats. In *Joint Army-Navy-NASA-Air Force*, 2010.

[91] W. D. Munter. Ontwerp, implementatie en testen van magnetische standcontrole voor cubesats. Master's thesis, KU Leuven, Belgium, 2015.

[92] NASA. Small spacecraft technologie state of the art. Technical report, NASA Ames Research Center, 2014.

[93] S. of Mathematics and Statistics. Distributing points on the sphere. `http://www.maths.unsw.edu.au/school/articles/me100.html`, 2010.

[94] E. Optics. *25mm, f/1.4 Compact Instrumentation Imaging Lens*, 2015.

[95] J. W. Percival, K. H. Nordsieck, and K. P. Jaehnig. The st5000: a high-precision star tracker and attitude determination system. In *Proc. SPIE 7010, Space Telescopes and Instrumentation 2008: Optical, Infrared and Millimeter*, 2008.

[96] Y. Perelman. *Astronomy for Entertainment*. University Press of the Pacific, Honolulu, Hawaii, 2000.

[97] A. Pignede. Detumbling of the ntnu test satellite. Master's thesis, Norwegian University of Science and Technology, 2014.

[98] B. M. Quine, V. Tarasyuk, H. Mebrahtu, and R. Hornsey. Determining star-image location: A new sub-pixel interpolation technique to process image centroids. *Computer Physics Communications*, 177(9):700–706, 2007.

[99] D. G. Ramírez. Optimizing the baffle of a cubesat star tracker using ray tracing. Master's thesis, KU Leuven, Belgium, 2015.

[100] M. Richard, B. Chamot, V. Gass, and C. Nicollier. Status of active debris removal (adr) developments at the swiss space center. In *IAF Symposium 2013*, 2013.

[101] Robert L. Staehle et al. Interplanetary cubesats: Opening the solar system to a broad community at lower cost. Technical report, NASA Jet Propulsion Laboratory, 2012.

[102] G. L. A. Rousseau, J. Bostel, and B. Mazari. New star pattern recognition algorithm for aps star tracker application: "oriented triangles". *IEEE Aerospace and Electronic Systems Magazine*, 20(2):27–31, 2005.

[103] G. Rufino and D. Accardo. Enhancement of the centroiding algorithm for star tracker measure refinement. *Acta Astronautica*, 53(2):135–147, 2002.

[104] A. Rush. Cubesats with a view require noaa licensing. `http://ipinspace.com/2012/10/10/cubesats-with-a-view-require-noaa-licensing/`, [retrieved 25 august 2015].

[105] M. Samaan, D. Mortari, J. L. Junkins, and et al. Recursive mode star identification algorithms. *Journal of IEEE transactions on aerospace and electronic systems*, 41(4):1246–1254, 2005.

[106] M. A. Samaan, D. Mortari, and J. L. Junkins. Nondimensional star identification for uncalibrated star cameras. *J. Astronaut. Sci.*, 54(1):95–111, 2006.

[107] T. Sasaki and M. Kosaka. A star identification method for satellite attitude determination using star sensors. In *15th International Symposium on Space Technology and Science*, volume 1, pages 1125–1130, 1986.

[108] M. S. Scholl. Star-field identification for autonomous attitude determination. *Journal of Guidance, Control, and Dynamics*, 18(1):61–65, 1995.

[109] M. M. Shara and M. D. Johnston. Artificial earth satellites crossing the fields of view of, and colliding with, orbiting space telescopes. *Publications of the Astronomical Society of the Pacific*, 98(606):814–820, 1986.

[110] B. Shucker. Ground-based prototype of cmos navigational star camera for small satellite applications. In 15^{th} *AIAA/USU conference on Small Satellites*, Utah State University, August 2001.

[111] M. D. Shuster. Approximate algorithms for fast optimal attitude computation. In *AIAA Guidance and Control Conference*, pages 88–95. AIAA New York, NY, August 1978.

[112] M. D. Shuster. The quest for better attitudes. *The Journal of Astronautical Sciences*, 54(3 -4):657–683, Jul-Dec. 2006.

[113] M. D. Shuster and D. C. Freesland. The statistics of taste and the inflight estimation of sensor precision. In *Flight Mechanics Symposium 2005*, 2005.

[114] M. D. Shuster and S. D. Oh. Three-axis attitude determination from vector observations. *Journal of Guidance and Control*, 4(1):70–77, 1981.

[115] M. J. Sidi. *Spacecraft Dynamics and Control: A Practical Engineering Approach*. Cambridge University Press, 1997. ISBN: 0-512-55072-6.

[116] G. L. Skrobot and R. Coelho. ELaNa - educational launch of nanosatellite providing routinge rideshare opportunities. In *Proceedings of the 26th Annual AIAA/USU Conference on Small Satellites*, 2012.

[117] C. P. SLO. Cubesat design specification rev 13. Technical report, California Polytechnic State University, 2015.

[118] M. W. Smith and et al. Exoplanetsat: detecting transiting exoplanets using a low-cost cubesat platform. In *Space Telescopes and Instrumentation 2010: Optical, Infrared, and Millimeter Wave*, 2010.

[119] D. Smitherman. Space elevators: An advanced earth-space infrastructure for the new millennium. Technical report, NASA Marshall space Flight Center, 2000.

[120] B. Spratling and D. Mortari. A survey on star identification algorithms. *Algorithms*, 2(1):93–107, 2009.

[121] A. Sterenharz. Piggyback launch opportunities for small spacecrafts to low earth orbit. In *SEMW-2010 - 1st International Conference, Vilnius*, 2010.

[122] H. Steyn. 3-axis attitude determination and control system for cubesats. Technical Report 1, ESL Electronic Systems Laboratory, July 2014.

[123] H. Steyn. Cubetorquer. Technical report, Innovus Technology Transfer, 2015.

[124] R. C. Stone. A comparison of digital centering algorithms. *The Astronomical Journal*, 97(4), April 1989.

[125] D. Torczynski, R. Amini, and P. Massioni. Magnetorquer based attitude control for a nanosatellite testplatform. In *infotech at Aerospace*, April 2010.

[126] C. Underwood and S. P. et al. Autonomous assembly of a reconfigurable space telescope (aarest) – a cubesat/microsatellite based technology demonstrator. In *Proceedings of the 27th Annual AIAA/USU Conference on Small Satellites*, 2013.

[127] D. A. Vallado, P. Crawford, and R. Hujsak. Revisiting spacetrack report nr3. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Keystone Colorado*, 2006.

[128] D. Vanderhenst. Design and implementation of a low-cost attitude determination and control system for cubesats. Master's thesis, KU Leuven, Celestijnenlaan 300B, 3001 Heverlee, Belgium, 2012.

[129] J. Vandersteen. *Observation and Estimation for Space Applications*. PhD thesis, KU Leuven, 2012.

[130] L. Visagie. Adcs interface control document version 3.0. Technical report, Universiteit Stellenbosch and University of Surrey, 2014.

[131] A. Vyas, M. Roopashree, and B. R. Prasad. Improved iteratively weighted centroiding for accurate spot detection in laser guide star based shack harmann sensor. In *Proceedings of SPIE Atmospheric and Oceanic Propagation of Electromagnetic Waves IV*, volume 7588, 2010.

[132] R. Walker, P. Galeone, H. page, A. Castro, F. Emma, N. Callens, and J. Ventura-Traveset. Esa hands-on space education project activities for university students: Attracting and training the next generation of space engineers. In *Education Engineering (EDUCON), 2010 IEEE*, 2010.

[133] J. R. Wertz. *Spacecraft Attitude Determination and Control*. D. Reidel Publishing Company, Dordrecht, Holland, 1978. ISBN 90-277-0959-9.

[134] J. R. Wertz and W. J. Larson. *Space Mission Analysis and Design*. Microcosm Press and Springer, Hawthorne, CA and New York NY, 3 edition, 1999.

[135] K. A. Winick. Cramér-rao lower bounds on the performance of charge-coupled-device optical position estimators. *Optical Society of America*, 3(11):1809–1815, 1986.

[136] XIMEA. *XiQ*. XIMEA GmbH Hansestraße 81 48165 Münster Germany, 2013.

[137] B. Yost and A. Petro. Cubesat proximity operations demonstration (cpod). Technical report, Nasa Space Technology Mission Directorate, 2013.

Curriculum

Tjorven Delabie received his MSc in Mechanical Engineering from KU Leuven University, Belgium in 2010. He started working on a Doctorate in engineering at the Department of Mechanical Engineering at KU Leuven in 2010. His research interests are attitude determination and control, star tracker algorithms and CubeSat development. He is the ADCS responsible for the SIMBA CubeSat of RMIB.

List of publications

Articles in internationally reviewed academic journals

Delabie, T., De Schutter, J., and Vandenbussch, B. (2015) Robustness and efficiency improvements for star tracker attitude estimation. *Journal of Guidance, Control and Dynamics*, 38 (11), 2108-2121.

Delabie, T., De Schutter, J., and Vandenbussche, B. (2015) An Accurate and Efficient Gaussian Fit Centroiding Algorithm for Star Trackers. *The Journal of the Astronautical Sciences*, 61 (1), 60-84.

Delabie, T., De Schutter, J., Vandenbussche, B. (2013). Highly Efficient Attitude Estimation Algorithm for Star Trackers Using Optimal Image Matching. *Journal of Guidance, Control and Dynamics*, 32 (6), 1672-1680.

Delabie, T., Durt, T., Vandersteen, J. (2013). Highly Robust Lost-in-space Algorithm Based On The Shortest Distance Transform. *Journal of Guidance, Control and Dynamics*, 36 (2), 476-484.

Papers at international scientific conferences and symposia, published in full in proceedings

Dewitte, S., Karatekin, O., Chevalier, A., Clerbaux, N., Meftah, M., Irbah, A., Delabie, T., (2015). The Sun-earth Imbalance Radiometer for a Direct Measurement of the Net Heating of the Earth, EGU General Assembly 2015, Apr 2015, Vienna, Austria

Delabie, T. (2015). Star Position Estimation Improvements for Accurate Star Tracker Attitude Estimation. Scitech2015. Kissimmee, FL, 03-09 January 2015 (pp. 1-15).

Delabie, T., Raskin, G., Vandenbussche, B., De Schutter, J. (2014). A Good Attitude Towards Improved Space Telescope Observations. SPIE Astronomical Telescopes + Instrumentation. Montreal, Canada, 22-27 June 2014 (pp. 1-17).

Dell'Elce, L., Kerschen, G., Delabie, T., Vandepitte, D., Fleury-Frenette, K., Lecat, J., Walewyns, T., Francis, L. (2013). Scientific and Technological Payloads Aboard the B3LSat CubeSat of the QB50 Network. IAA Conference on University Satellite Missions and CubeSat Workshop. Rome, Italy, 3-9 February 2013.

Delabie, T. (2013). Robust and Efficient Star Tracker Attitude Estimation Using the AIM Algorithm. *Proceedings of the AIAA Guidance, Navigation and Control Conference*. AIAA Guidance, Navigation and Control Conference. Boston, 19-22 August 2013 (pp. 1-16).

Delabie, T., De Schutter, J., Vandenbussche, B. (2013). An Accurate and Efficient Gaussian Fit Centroiding Algorithm for Star Trackers. AAS/AIAA Space Flight Mechanics Meeting. Kauai, HI, 10-14 February 2013.

Delabie, T., Vandersteen, J., Vandepitte, D. (2012). Low-cost Attitude Determination and Control System Using Reaction Wheels and Star Tracker. *Proceedings of the Small Satellites Systems & Services Conference 2012*. Small Satellites Systems & Services. Portoroz, Slovenia, 04-08 June 2012.

Delabie, T. (2012). A Highly Efficient Attitude Estimation Algorithm for Star Trackers Based on Optimal Image Matching. *Proceedings of the AIAA Guidance, Navigation, and Control Conference*: Vol. 1 (1). AIAA Guidance, Navigation, and Control Conference. Minneapolis, MN, 13-16 August 2012 (pp. 1-16).

Delabie, T., Durt, T., Vandersteen, J. (2011). A Highly Robust Lost In Space Algorithm Based on the Shortest Distance Transform. *Proceedings of the AIAA Guidance, Navigation and Control Conference*. AIAA Guidance, Navigation and Control Conference. Portland, Or 8-11 August 2011.

Meeting abstracts, presented at international scientific conferences and symposia, published or not published in proceedings or journals

Gómez Ramírez, D., Boucher, M., Delabie, T., Raskin, G., Vandepitte, D. (2015). Optimization of a CubeSat Baffle Using Ray Tracing. Workshop on Innovative Technologies for Space Optics, Noordwijk, NL, 26 November 2015.

Delabie, T., De Schutter, J., Vandenbussche, B. (2014). Star Tracker Cost Reduction for Small Satellites. European CubeSat Symposium. Estavayer-le-lac, CH, 14-16 October 2014.

Delabie, T., Vandepitte, D., Vandenbussche, B., De Schutter, J. (2013). Low-Cost Star Tracker with Highly Efficient Algorithms. European CubeSat Symposium. Brussels, 3-5 June 2013.

Delabie, T., Vandersteen, J., Vandepitte, D. (2012). Low-Cost Attitude Determination and Control System using Reaction Wheels and Star Tracker. European CubeSat Symposium. Brussels, 30/1 - 2/2 2012.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF MECHANICS
SPACE RESEARCH
Celestijnenlaan 300B box 2402
B-3001 Heverlee
tjorven.delabie@kuleuven.be
http://mech.kuleuven.be