



UNIVERSITEIT • STELLENBOSCH • UNIVERSITY
jou kennisvennoot • your knowledge partner

Autonomous Docking for a Satellite Pair Using Monocular Vision

by

Dewald Mienie



*Thesis presented at the University of Stellenbosch in
partial fulfilment of the requirements for the degree of*

Masters of Engineering

Department of Electrical Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Study leader: Prof W.H. Steyn

November 2008

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

D. Mienie

Date:

Abstract

Autonomous rendezvous and docking is seen as an enabling technology. It allows, among others, the construction of larger space platforms in-orbit and also provides a means for the in-orbit servicing of space vehicles.

In this thesis a docking sequence is proposed and tested in both simulation and practice. This therefore also requires the design and construction of a test platform. A model hovercraft is used to emulate the chaser satellite in a 2-dimensional plane as it moves relatively frictionlessly. The hovercraft is also equipped with a single camera (monocular vision) that is used as the main sensor to estimate the target's pose (relative position and orientation). An imitation of a target satellite was made and equipped with light markers that are used by the chaser's camera sensor.

The position of the target's lights in the image is used to determine the target's pose using a modified version of Malan's Extended Kalman Filter [20]. This information is then used during the docking sequence.

This thesis successfully demonstrated the autonomous and reliable identification of the target's lights in the image, and the autonomous docking of a satellite pair using monocular camera vision in both simulation and emulation.

Opsomming

Die tegnologie om twee satelliete outonoom aan mekaar te kan koppel in die ruimte word as 'n bemagtigende tegnologie gesien. Dit maak dit moontlik om, onder andere, groter ruimte platforms te konstrueer in die ruimte en verskaf ook 'n manier om ruimtevoertuie te onderhou in die ruimte.

In hierdie tesis word 'n koppelingsekwensie voorgestel en getoets in beide simulاسie en in die praktyk. Dus word die ontwerp en konstruksie van 'n toetsplatform genoodsaak. 'n Model skeertuig ("hovercraft") word gebruik om die volger satelliet te emuleer in 'n 2-dimensionele vlak, aangesien dit redelik wrywingloos beweeg. Die skeertuig is ook toegerus met 'n kamera (monokulêre visie) wat dien as die hoofsensor vir die afskatting van die teiken satelliet se relatiewe posisie en oriëntasie. 'n Nabootsing van die teikensatelliet was gemaak en toegerus met ligte wat gebruik word deur die volger se kamera.

Die posisie van die teiken se ligte in die kamerabeeld word gebruik om die teiken se relatiewe posisie en oriëntasie af te skat deur gebruik te maak van aangepasde weergawe van Malan [20] se Uitgebreide Kalman Filter. Hierdie inligting word dan gebruik om die koppelingsekwensie mee uit te voer.

Hierdie tesis het die betroubare en outonome herkenning van die teiken se ligte in die kamerabeeld en die outonome koppeling van 'n satelliet paar, deur gebruik te maak van monokulêre kameravise, suksesvol gedemonstreer in beide simulاسie en emulasie toestande.

Acknowledgements

The author would like to thank the following people for their contribution towards this project:

- I would above all like to thank my parents for the opportunity to further my studies at the University of Stellenbosch and for their continuous support throughout the years.
- I would also like to thank my study leader, Prof Steyn, for all his support.
- For his help with debugging the camera, I would like to thank Xandri Farr from Sun-Space.
- Thank you to the Wilhelm Frank Bursary for providing me with the means to do my MScEng.
- Also, many thanks to Mr. Wessel Croukamp at SMD for all his patience and help with the mechanical modifications that were done on the hovercraft and the design of the camera clamp.
- Thank you to the students in the ESL, especially Bernard Visser and Ruan de Hart, for their contributions to this thesis. Also thanks to Gerrit de Villiers for proofreading this thesis.
- Lastly, but definitely not the least, thank you to all my friends for their friendship, support and laughter.

Contents

Declaration	i
Abstract	ii
Opsomming	iii
Acknowledgements	iv
Contents	v
Abbreviations	ix
List of Figures	xi
List of Tables	xvi
Nomenclature	xvii
1 Introduction	1
1.1 Background	1
1.2 Thesis Goals	3
1.3 Approach	3
1.4 Thesis Layout	3
1.5 Achievements	4
2 The Docking Sequence	5
2.1 A Few Definitions	5
2.2 Overview of the Docking Sequence	7
2.3 Detailed Description	7
2.4 Guidance System	8
3 Using a Monocular Camera	10
3.1 Basic Principle	10
3.2 Monocular Vision versus Stereo Vision	11
3.3 Methods Considered	11
3.4 Requirements	12

4	Light Model	13
4.1	Definitions	13
4.2	Light Radiation Pattern	14
4.3	Pinhole Camera Model	14
4.4	The Light Model	15
4.5	Conclusion	18
5	Identification and Tracking of Light Centroids	19
5.1	Overview	19
5.2	Movement of the Target Satellite's Centroids and the Star's Centroids in the Image	20
5.3	The Matching Algorithm	25
5.4	The Identification and Tracking Unit	26
5.5	Searching for Centroids in the Image	30
5.6	Rapid Centroid Determination Algorithm	31
6	System Overview	34
6.1	Chaser Overview	34
6.2	Choice of Operating System for the Chaser's OBC	35
6.3	Target Overview	36
6.4	Selection of the Sample Frequencies for the Inner and Outer Loop Controllers . .	36
6.5	Conclusion	37
7	Hardware Implementation	38
7.1	The Onboard Computer (OBC)	38
7.2	Navigation and Attitude Sensors	38
7.3	The Accelerometers and Gyros	40
7.4	Actuators	52
7.5	Motor Driver Board	57
7.6	Testing of the RGPS and Results	58
8	The Camera Sensor	61
8.1	Description of the Camera Module and Camera Axis	61
8.2	Camera Clamp	63
8.3	Lens Distortion	63
8.4	Camera Calibration	65
8.5	The Calibration Procedure in Detail	66
8.6	Results from the Calibration Procedure	72
9	Tailoring DF Malan's EKF	75
9.1	Preface	75
9.2	Using Euler Angles	76
9.3	The Projective Camera Model in Matrix Form	78
9.4	The Customised Filter	80
9.5	The Noise Variances	82

9.6	Conclusion	83
10	Inner Loop Controller	84
10.1	Simulation and Controller Design Methodology	84
10.2	The Model Used to Design the Controller	85
10.3	Controller and Estimator Design	90
10.4	Simulation Results	92
10.5	Practical Results	95
10.6	Velocity Sensor and Controllers	98
10.7	Conclusion	103
11	The System EKF	104
11.1	The State Vector	104
11.2	Propagation	104
11.3	Measurement Update	105
11.4	Compensating for the Measurement Delay	106
11.5	Noise Variances	107
11.6	Conclusion	108
12	Outer Loop Controller And Guidance System	109
12.1	Overview of the Outer Loop Controllers	109
12.2	The Guidance System	109
12.3	Models Used to Design the Outer Loop Controllers	111
12.4	The Outer Loop Controllers	114
12.5	Conclusion	119
13	Simulation of the Docking Sequence	120
13.1	Overview	120
13.2	Simulation Parameters	121
13.3	Simulation Results	122
13.4	Conclusion	130
14	Practical Results	131
14.1	The Test Setup	131
14.2	Conclusion	140
15	Conclusion and Recommendations	141
15.1	Conclusions	141
15.2	Recommendations	143
	Bibliography	145
	Appendices	148
A	System Block Diagrams	149

B	Derivation of Formulas in Chapter 5	157
B.1	Movement of stars' centroids due to the rotation and translation of the satellites around one another and the earth	157
B.2	Movement of stars' centroids due to the rotation of the chaser satellite around its own axis	159
B.3	Movement of the target's lights' centroids due to the relative translation between the satellites	160
C	Rapid Centroid Determination Algorithm	161
C.1	Some Background	161
C.2	The Code	162
D	Photos of the Hardware	167
D.1	The Chaser and Target	167
D.2	The Docking Mechanisms	169
D.3	The Camera Unit	170
D.4	The Velocity Sensor	170
E	Homogeneous Vectors and the Projective Camera Matrix	171
E.1	Basic Knowledge Required	171
E.2	Using the Projective Camera Matrix	172
E.3	Inserting a DCM and Translation	173
F	Hovercraft Model	174
F.1	The Simulink Hovercraft Block	174
F.2	Using the Hovercraft Block	175
F.3	The Inside of the Hovercraft Block	176
F.4	Conclusion	177
G	Modeling Of Mechanical Vibration	179
G.1	The Model	179
G.2	Conclusion	180

Abbreviations

- ADC - Analog to Digital Converter
- CDPGPS - Carrier-Differential Phase GPS
- CG - Centre Of Gravity
- COD - Centre Of Distortion
- CPU - Central Processing Unit
- DCM - Direction Cosine Matrix
- ECEF - Earth Centred, Earth Fixed
- EKF - Extended Kalman Filter
- ESC - Electronic Speed Control
- ESL - Electronic Systems Laboratory
- ETS - Engineering Test Satellite
- GLONASS - Global Orbiting Navigation Satellite System
- GPS - Global Positioning System
- ISS - International Space Station
- LQR - Linear Quadratic Regulator
- MSDOS - Microsoft Disk Operating System
- NASDA - National Space Development Agency of Japan
- OBC - Onboard Computer
- PWM - Pulse Width Modulation
- RAM - Random Access Memory
- RF - Radio Frequency
- RGPS - Relative GPS

- RVD - Rendezvous and Docking
- SIFT - Scale Invariant Feature Transform
- US - United States
- WOI - Window Of Interest

List of Figures

1.1	The emulation setup. Note that in this photo the black cloth is not fully tensioned.	4
2.1	Parameter and axis definitions.	5
2.2	Phase definitions. The left satellite is the (T)arget and the right satellite is the (C)haser.	6
2.3	Target's approach corridor and keep out zone.	9
2.4	Conceptual representation of the guidance system (Adapted from [6]).	9
3.1	Stereo vision.	10
3.2	Monocular vision.	11
4.1	Light model.	14
4.2	A light radiation pattern.	14
4.3	Pinhole camera model.	15
5.1	Breakdown of the identification and tracking process.	20
5.2	Movement of the chaser and target around each other and the earth.	20
5.3	The star's centroid move from its old position (circle) to its new position (triangle), while the target satellite (square) remains stationary in the image.	21
5.4	Movement of star's centroid due to the rotation of the chaser satellite (C) around its own axis.	22
5.5	Movement of the target's lights' centroids due to the relative rotation of the satellites.	22
5.6	Movement of the target's light's centroid due to the rotation of the chaser around its own axis.	23
5.7	As the chaser moves closer to the target (a), the target's lights' centroids move away from each other in the image (b).	24
5.8	Relative translation of the chaser along another axis.	24
5.9	Matching old and new centroids.	25
5.10	The gradual appearance and disappearance of centroids.	30
5.11	The light radiation pattern used in this thesis.	30
5.12	A $n \times n$ square (shaded) with bordering pixels (clear).	31
5.13	The ratio, η , between the region growing algorithm's and the new algorithm's execution speed.	32
5.14	For the light spot shown in (a), the faster algorithm will only use the shaded region shown in (b) to determine the centroid.	33

6.1	Overview of the chaser.	34
6.2	Overview of the target.	36
7.1	The result of the integration process.	44
7.2	The Allan deviation for the x-axis accelerometer.	45
7.3	The Allan deviation for the y-axis accelerometer.	45
7.4	The Allan deviation for the z-axis rate gyro.	47
7.5	Full accelerometer model. (X-axis accelerometer shown.)	48
7.6	Full gyro model. (Note: The input angular rate is in rad.s^{-1})	49
7.7	The sensor board.	49
7.8	Motor model.	53
7.9	Actuator calibration setup.	53
7.10	Direction of positive force value.	54
7.11	Fan type 1 transfer curve.	54
7.12	Fan type 2 transfer curve.	55
7.13	The fan currents. (a) Fan 1's current. (b) Fan 2's current.	55
7.14	Linearising the actuator. (a) The non-linear fan with the inverse transfer curve. (b) The linearised fan model.	57
7.15	Drift of the relative distance between the GPS receivers along the North- and East-axis.	59
7.16	Drift of the relative distance between the GPS receivers.	59
7.17	The integral of the velocity along the North-axis.	60
7.18	The integral of the velocity along the East-axis.	60
8.1	The camera unit.	61
8.2	Simplified view of the sensor and lens unit's construction.	62
8.3	(a) The sensor. (b) Sensor axis.	62
8.4	(a) The camera sensor unit. (b) The camera axis.	62
8.5	The camera clamp.	63
8.6	Radial Distortion. The dashed line represents the rectangular object as it would appear in the absence of radial distortion and solid line shows the object shape in the presence of (a) barrel and (b) pincushion distortion. Figure taken from [24].	64
8.7	The test grid used.	66
8.8	Mounting the camera in the clamp.	66
8.9	Mount the camera axis orthogonally with respect to the test pattern's axis.	67
8.10	The test object's axis orientation in the camera axis.	68
8.11	The photo used for calibration.	69
8.12	(a) Isolating the centroids locations. (b) The result of local thresholding.	69
8.13	The final processed image containing the centroids.	70
8.14	The basic concept used to determine the camera's parameters.	70
8.15	The mathematical photo (white points) superimposed on the actual photo (black spots).	73
8.16	The pixels used during the correction process. (The black area in the centre of the image.)	73

8.17	The original photo.	74
8.18	The corrected picture.	74
9.1	All the relevant axes.	76
9.2	The Euler axis, $\hat{\mathbf{e}}$, and Euler angle, θ_e . (Picture adapted from Wikipedia)	77
10.1	Chaser's Body Fixed Reference Axis.	85
10.2	Basic Hovercraft Model.	86
10.3	The inner loop simulation.	93
10.4	Acceleration of the CG along the x-axis.	93
10.5	Acceleration of the CG along the y-axis.	94
10.6	Angular rate around z-axis.	94
10.7	The estimated X-acceleration (black line) and the measured X-acceleration (gray line).	95
10.8	The tilting as result of the actuator forces.	96
10.9	The estimated Y-acceleration (black line) and the measured Y-acceleration (gray line).	97
10.10	The estimated angular rate (black line) closely follows the measured angular rate (gray line).	98
10.11	The velocity sensor.	99
10.12	Velocity of the CG along the x-axis.	101
10.13	Velocity of the CG along the y-axis.	101
10.14	The estimated X-velocity (black line) and the measured X-velocity (gray line).	102
10.15	The estimated Y-velocity (black line) and the measured Y-velocity (gray line).	103
11.1	Processing the camera measurement.	105
11.2	The measurement delay as seen by the system EKF.	107
12.1	The regions where each controller set is used.	110
12.2	The 2nd order systems used to approximate the closed loop system consisting out of the inner loop controllers and plant.	111
12.3	Diagram used to determine the conversion equations.	112
12.4	The 2nd order models encapsulated by the non-linear conversion blocks.	112
12.5	Under the assumption that $\alpha \approx 0$, the models are decoupled.	113
12.6	The decoupled models used to design the outer loop controllers.	113
12.7	Model used to design the far controller for α	114
12.8	Simplified model used to design the far controller for α	114
12.9	The far controller for the angle α	115
12.10	Model used to design the close controller for β	116
12.11	Simplified model used to design the close controller for β	116
12.12	Simplified model used to design the close controller for α	118
13.1	Graphical representation of the software layers.	120
13.2	The chaser's trajectory (gray line) in the simulation. The direction of the chaser's positive x-axis is given by the black arrows.	122
13.3	Target's estimated (black) and actual (gray) position along the camera's z-axis.	124

13.4	Target's estimated (black) and actual (gray) velocity along the camera's z-axis.	124
13.5	Target's estimated (black) and actual (gray) position along the camera's y-axis.	125
13.6	Target's estimated (black) and actual (gray) velocity along the camera's y-axis.	125
13.7	Target's estimated (black) and actual (gray) orientation with respect to the chaser.	126
13.8	Target's estimated (black) and actual (gray) angular velocity with respect to the chaser's angular velocity.	126
13.9	The estimated (black) and actual (gray) distance r	128
13.10	The estimated (black) and actual (gray) angle α	128
13.11	The estimated (black) and actual (gray) angle β	129
13.12	Target's estimated (black) and actual (gray) angular rate around its own z-axis.	129
14.1	The test setup.	131
14.2	The target's estimated position along the camera's z-axis.	134
14.3	The target's estimated velocity along the camera's z-axis.	134
14.4	The target's estimated position along the camera's y-axis.	135
14.5	The target's estimated velocity along the camera's y-axis.	135
14.6	The target's estimated orientation with respect to the chaser.	136
14.7	Target's estimated angular velocity with respect to the chaser's angular velocity.	136
14.8	The estimated distance r	138
14.9	The estimated angle α	138
14.10	The estimated angle β	139
14.11	Target's estimated angular rate around its own z-axis.	139
15.1	Autonomous docking test bed used by Romano. (Picture taken from [27].)	143
A.1	Overview 1.	150
A.2	Overview 2.	151
A.3	Process Inner Loop data.	152
A.4	Camera Operation.	153
A.5	Initialise Camera.	154
A.6	Track Centroids.	155
A.7	Use Model.	156
B.1	Movement of a star's centroid due to chaser's and target's rotation around each other and the earth.	157
B.2	Movement of star's centroid due to the rotation of the chaser satellite around its own axis.	159
B.3	Movement of the target's lights' centroids due to the relative translation between the satellites.	160
D.1	The chaser.	167
D.2	Another view of the chaser.	168
D.3	The target.	168
D.4	The pin is the chaser's docking mechanism.	169

D.5 The hole is the target’s docking mechanism. 169

D.6 The camera unit. 170

D.7 The velocity sensor. 170

F.1 Hovercraft block. 174

F.2 Using the hovercraft block. 175

F.3 The inside of the hovercraft block. 178

G.1 The vibration model. 179

List of Tables

7.1	The results from the calibration experiments.	42
7.2	Results of the 3rd gyro calibration test.	44
7.3	Noise parameters of the accelerometers.	46
7.4	The standard deviations of the accelerometers' noise components.	46
7.5	Noise parameters of the gyros.	47
7.6	The standard deviations of the gyro's noise components.	47
7.7	Comparing the results from the two different methods.	49
8.1	Calibration Results.	72
10.1	The hovercraft's parameters.	89
13.1	The RMS errors of Malan's EKF.	123
13.2	The RMS errors of system EKF.	127

Nomenclature

Scalars:

f	The camera lens' focal length.
f_c	Cut-off frequency.
f_{s_i}	Inner loop sample frequency.
f_{s_o}	Outer loop sample frequency.
i	Current.
g	Gravitational acceleration (9.81 m.s^2)
k_1, k_2	Distortion model parameters.
m	Hovercraft's mass.
n_{duty1}, n_{duty2}	Duty cycle value for actuator (fan) type 1 and 2, respectively.
n_Y	Disturbance acceleration along the camera's y-axis.
n_Z	Disturbance acceleration along the camera's z-axis.
n_ω	Disturbance angular acceleration around the camera's x-axis.
p	Size of a pixel.
p_X	The size of a pixel along the CMOS sensor's x-axis.
p_Y	The size of a pixel along the CMOS sensor's y-axis.
r	Distance between chaser's body axis origin, O_C , and the target's body axis origin, O_T .
\bar{r}	The distance between the chaser's and target's axes' origins as estimated by the system EKF.
r_d	Distorted radius.
r_u	Undistorted radius.
r_{FAR}	The distance between the chaser's and target's body axis origin so that the target becomes visible to the chaser.
$r_{RETREAT}$	The distance to which the chaser satellite retreats if any of the safety conditions are not met.
r_{SAFE}	The radius of the sphere (the target's safety sphere) that encapsulates the whole target satellite with all its external components.
t	Time.

u_0	X-coordinate of the principle point in the CMOS sensor axis, in pixel units.
u_r	Chaser's commanded velocity along the target's radial axis.
u_{Cz}	Angular rate commanded around the chaser's z-axis.
u_{CT}	Chaser's commanded angular rate around the target.
u_X	Velocity commanded along the chaser's x-axis.
u_Y	Velocity commanded along the chaser's y-axis.
v_0	Y-coordinate of the principle point in the CMOS sensor axis, in pixel units.
v_r	Chaser's measured velocity along the target's radial axis.
\bar{v}_y	Velocity of the target along the camera's y-axis as estimated by Malan's EKF.
\bar{v}_z	Velocity of the target along the camera's z-axis as estimated by Malan's EKF.
v_X	Velocity measurement along the chaser's x-axis.
v_Y	Velocity measurement along the chaser's y-axis.
x_P^{CAM}	X-coordinate of a point P in the camera axis.
y_P^{CAM}	Y-coordinate of a point P in the camera axis.
z_P^{CAM}	Z-coordinate of a point P in the camera axis.
x_d, y_d	Distorted pixel's x- and y-coordinate.
x_{sensor}, y_{sensor}	X- and Y-coordinate of a point in the CMOS sensor axis.
$\bar{x}_{sensor}, \bar{y}_{sensor}$	Estimated x- and y-coordinate of a light in the image.
x_u, y_u	Undistorted pixel's x- and y-coordinate.
x_{COD}, y_{COD}	X- and Y-coordinate of the centre of distortion.
\ddot{x}_m, \ddot{y}_m	Measured acceleration along the sensor board's x- and y-axis.
$\ddot{x}_{AD}, \ddot{y}_{AD}$	The acceleration along the sensor board's x- and y-axis in ADC units.
$\ddot{x}_{AD}^+, \ddot{y}_{AD}^+$	The acceleration along the sensor board's x- and y-axis in ADC units when the gravitational acceleration vector coincides with the positive x- and y-axis, respectively.
$\ddot{x}_{AD}^-, \ddot{y}_{AD}^-$	The acceleration along the sensor board's x- and y-axis in ADC units when the gravitational acceleration vector coincides with the negative x- and y-axis, respectively.
$\ddot{x}_{CG}, \ddot{y}_{CG}$	Acceleration of the chaser's CG along its x- and y-axis, respectively.
\bar{y}	Position of the target along the camera's y-axis as estimated by Malan's EKF.
\bar{z}	Position of the target along the camera's z-axis as estimated by Malan's EKF.
D_1, D_2	Accelerometers' offsets along the sensor board's x- and y-axis.
D_3	Rate gyro's offset.
F_1, F_2	Output force of actuator (fan) type 1 and 2, respectively.
F_{IL}	Magnitude of the input force vector of the fan on the left hand side (-y) of the hovercraft.

F_{IR}	Magnitude of the input force vector of the fan on the right hand side (+y) of the hovercraft.
F_{IS}	Magnitude of the input force vector of the side fan on the hovercraft.
\hat{F}_{IX}	Magnitude of the input virtual force vector along the chaser's x-axis.
\hat{F}_{IY}	Magnitude of the input virtual force vector along the chaser's y-axis.
F_{OL}	Magnitude of the output force vector of the fan on the left hand side (-y) of the hovercraft.
F_{OR}	Magnitude of the output force vector of the fan on the right hand side (+y) of the hovercraft.
F_{OS}	Magnitude of the output force vector of the side fan on the hovercraft.
\hat{F}_{OX}	Magnitude of the output virtual force vector along the chaser's x-axis.
\hat{F}_{OY}	Magnitude of the output virtual force vector along the chaser's y-axis.
\hat{F}_X	Magnitude of the virtual force vector along the chaser's x-axis.
\hat{F}_Y	Magnitude of the virtual force vector along the chaser's y-axis.
G_{acc}	Theoretical gain from the measured acceleration to the ADC's output.
G_{gyro}	Theoretical gain from the measured angular to the ADC's output.
I_{Light_n}	Intensity of light n .
I_{ZZ}	Hovercraft's moment of inertia around its z-axis.
K	The rate random walk spectral density coefficient.
K_x, K_y	Calibration constants for the sensor board's x- and y-axis.
K_θ	Calibration constants for the sensor board's z-axis.
Q	The velocity/angle random walk spectral density coefficient.
\hat{T}	Magnitude of the virtual torque around the chaser's z-axis.
T_{s_i}	Inner loop sample period.
T_{s_o}	Outer loop sample period.
\hat{T}_I	Magnitude of the input virtual torque around the chaser's z-axis.
\hat{T}_O	Magnitude of the output virtual torque around the chaser's z-axis.
X	Position of the target along the camera's x-axis. (A constant.)
α	Angle between the chaser's x-axis and the line $\overline{O_C O_T}$.
$\bar{\alpha}$	The angle α as estimated by the system EKF.
β	Angle between the target's x-axis and the line $\overline{O_C O_T}$.
$\bar{\beta}$	The angle β as estimated by the system EKF.
γ_n	Viewing angle of light n .
ϵ_α	Maximum allowable error in α during the final approach.
ϵ_β	Maximum allowable error in β during the final approach.
θ_{AD}	Angle in ADC units.

θ_{Cz}	Angle through which the chaser satellite turned between 2 successive photos.
$\theta_{max}, \theta_{min}$	Maximum and minimum angle, respectively.
$\dot{\theta}$	Angular velocity of the chaser around its z-axis.
$\dot{\theta}_{AD}$	The angular rate around the sensor board's z-axis in ADC units.
$\dot{\theta}_m$	The measured angular rate around the sensor board's z-axis.
$\ddot{\theta}$	Angular acceleration of the chaser around its z-axis.
σ_Y	Standard deviation of the disturbance acceleration along the camera's y-axis.
σ_Z	Standard deviation of the disturbance acceleration along the camera's z-axis.
σ_ω	Standard deviation of the disturbance angular acceleration around the camera's x-axis.
σ_{ARW}^2	Angular rate random walk covariance.
σ_{RRW}^2	Rate random walk covariance.
σ_{VRW}^2	Velocity random walk covariance.
τ_{fan1}, τ_{fan2}	Mechanical time constant of actuator (fan) type 1 and 2, respectively.
τ_L, τ_R, τ_S	Time constants of the left, right and sideways fans, respectively.
τ_X, τ_Y, τ_Z	Time constants of the virtual forces along the x- and y-axis and the virtual torque around the z-axis, respectively.
ϕ, θ, ψ	Angle of rotation around the x-, y- and z-axis, respectively.
$\bar{\psi}$	Rotation of the target's body axis with respect to the chaser's body axis as estimated by Malan's EKF.
ω_{C_T}	Angular rate of the chaser around the target.
ω_{C_Z}	Angular rate of the chaser around its own z-axis.
$\bar{\omega}_{C_Z}$	The angular rate of the chaser around its z-axis as estimated by the system EKF.
$\overline{\omega_{C_Z}}$	The average angular rate of the chaser around its own z-axis between the current and previous photo.
ω_{T_Z}	Angular rate of the target around its own z-axis.
$\bar{\omega}_{T_Z/C_Z}$	The angular rate of the target's body axis with respect to the chaser's body axis as estimated by Malan's EKF.
$\Delta\dot{x}_{AD}, \Delta\dot{y}_{AD}$	Difference between \dot{x}_{AD}^+ and \dot{x}_{AD}^- , and \dot{y}_{AD}^+ and \dot{y}_{AD}^- , respectively.
$\Delta N, \Delta E$	Distance between 2 GPS receivers along the North- and East-axis.
$\overline{\Delta\theta}$	Average angular difference.

Vectors:

$\mathbf{e}_{L_n}^{TGT}$	Unit vector giving the direction of the light radiation pattern's symmetry axis of the light n in the target satellite's axis.
$\mathbf{i}, \mathbf{j}, \mathbf{k}$	Unit vectors along the camera axis' x-, y- and z-axis, respectively.
$\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$	Unit vectors along the rotated camera axis' x-, y- and z-axis, respectively.

$\tilde{\mathbf{m}}_j$	Positions of the projection of a test pattern's point j in the image as a homogeneous vector.
$\mathbf{r}_{L_n}^{CAM}$	Position of the light n in the camera axis.
\mathbf{r}_{TGT}^{CAM}	Position of the target satellite axis' origin in the camera axis.
$\mathbf{r}_{L_n}^{TGT}$	Position of the light n in the target satellite's axis.
$\hat{\mathbf{r}}_{L_n}$	Rotated position vector of a light n in the camera axis.
\mathbf{r}_{OL}	Position of \mathbf{F}_{OL} in the chaser's reference axis.
\mathbf{r}_{OR}	Position of \mathbf{F}_{OR} in the chaser's reference axis.
\mathbf{r}_{OS}	Position of \mathbf{F}_{OS} in the chaser's reference axis.
\mathbf{r}_{CG}	Position of the chaser's CG in the chaser's reference axis.
$\mathbf{v}_1, \mathbf{v}_2$	Measurement noise vectors.
$\mathbf{w}_1, \mathbf{w}_2$	Process noise vectors.
$\bar{\mathbf{x}}_{sys}$	System EKF's state vector.
$\bar{\mathbf{x}}_{DF}$	Malan's EKF's state vector.
$\ddot{\mathbf{x}}_{ref}$	Reference acceleration in the chaser's body axis. It is used instead of $\ddot{\mathbf{x}}_{ref}^{CHR}$ to simplify notation.
$\ddot{\mathbf{x}}_{ref}^{CHR}$	Reference acceleration in the chaser's body axis. To simplify notation it is also referred to as $\ddot{\mathbf{x}}_{ref}$.
$\ddot{\mathbf{x}}_{AD}$	Measured acceleration in the sensor board's axis in ADC units.
\mathbf{F}_{OL}	Output force vector of the fan on the left hand side (-y) of the hovercraft.
\mathbf{F}_{OR}	Output force vector of the fan on the right hand side (+y) of the hovercraft.
\mathbf{F}_{OS}	Output force vector of the side fan on the hovercraft.
$\tilde{\mathbf{M}}_j$	Position of a test point j in the test object's axis as a homogeneous vector.
$\Delta\ddot{\mathbf{x}}_{ref}$	Difference in reference accelerations in the chaser's body axis.
$\Delta\ddot{\mathbf{x}}_{AD}$	Difference in measured accelerations (in the sensor board's axis) in ADC units.

Matrices:

\mathbf{C}	Cross coupling matrix for the accelerometers.
\mathbf{Q}_{sys}	Continuous process noise covariance matrix for system EKF.
\mathbf{Q}_{sys_k}	Discrete equivalent of the continuous process noise covariance matrix for system EKF.
\mathbf{Q}_{DF}	Continuous process noise covariance matrix for Malan's EKF.
\mathbf{Q}_{DF_k}	Discrete equivalent of the continuous process noise covariance matrix for Malan's EKF.
\mathbf{R}_{sys_k}	Measurement noise covariance matrix for system EKF.

\mathbf{R}_{DE_k}	Measurement noise covariance matrix for Malan's EKF.
\mathbf{T}_p	Partial DCM.
$\mathbf{T}_{CAM \rightarrow TGT}$	DCM that gives the target's body axis orientation with respect to the camera axis.
$\mathbf{T}_{CHR \rightarrow TGT}$	DCM that gives the target's body axis orientation with respect to the chaser's body axis.
$\mathbf{T}_{CHR \rightarrow CAM}$	DCM that gives the camera axis orientation with respect to the chaser's body axis.
$\mathbf{T}_{CHR \rightarrow SB}$	DCM that gives the sensor board axis orientation with respect to the chaser's body axis.
$\Delta \ddot{\mathbf{X}}_{ref}$	Matrix where each column is the difference in reference accelerations in the chaser's body axis.
$\Delta \ddot{\mathbf{X}}_{AD}$	Matrix where each column is the difference in measured accelerations (in the sensor board's axis) in ADC units.

Notation:

O_C	Origin of the chaser's body axis.
O_T	Origin of the target's body axis.
$\overline{O_C O_T}$	Line between points O_C and O_T .
$ s $	The absolute value of a scalar s .
$\ \mathbf{v}\ $	The Euclidean norm of vector \mathbf{v} .
$\mathbf{a} \cdot \mathbf{b}$	The dot product of vectors \mathbf{a} and \mathbf{b} .
$\text{proj}_{\mathbf{a}} \mathbf{b}$	The projection of vector \mathbf{b} on vector \mathbf{a} .
$ \mathbf{A} $	The determinant of matrix \mathbf{A} .
\mathbf{A}^T	The transpose of matrix \mathbf{A} .

Chapter 1

Introduction

1.1 Background

Spacecraft rendezvous and docking dates back as far as the late 1960s. Examples of this includes the US Gemini and Apollo missions and the unmanned Russian Cosmos missions. [27]

Since the early 1980s large amounts of time and money has been invested in researching and developing the technology required for the autonomous rendezvous and docking (RVD) of spacecraft. [6] Since then there have been several successful RVD missions.

In 1998, the National Space Development Agency of Japan (NASDA) successfully demonstrated the fully autonomous rendezvous and docking of two unmanned spacecraft. These two satellites were collectively known as Engineering Test Satellite-VII (ETS-VII). [14][27]

Soon thereafter, also in 1998, construction on the International Space Station (ISS) started.

There are many reasons why RVD technology is so sought after and considered as a necessary stepping stone for future space missions. Some of these reasons are: [4][6][14][18][21][25][27][29]

- It allows the assembly of larger space platforms, such as the ISS, in orbit. This in turn provides the means for deeper space exploration.
- Logistic support for these space platforms. (Crew exchanges and re-supply.)
- In orbit servicing. The ability to repair and maintain space vehicles in orbit is of great advantage for many parties. The main advantage is the reduction of expenses for satellite operators and manufacturers. It is not uncommon that a satellite becomes unusable simply because it has exhausted its fuel or batteries or did not deploy correctly. Typically, these repairs have to be done by an astronaut and are very expensive due to the risks involved. Therefore the life of a satellite (and other space vehicles) can be increased drastically if in orbit servicing is possible.
- Retrieval. This includes the capture and return of space vehicles as well as the return of samples from other bodies in the Solar System.

The Electronic Systems Laboratory (ESL) at the University of Stellenbosch is interested in RVD technology as it might be used in potential future projects such as:

- Formation flight. The idea is to have a 2 satellite formation where the smaller satellite, typically a nanosatellite, will not have solar panels to reduce cost. It will draw all its power from onboard batteries and must therefore periodically dock at the larger satellite (with solar panels) to recharge its batteries.
- For an inspection satellite. An inspection satellite, equipped with a camera, will be attached to the main satellite. When a visual inspection of the main satellite has to be performed, the inspection satellite will be deployed. After the inspection is completed, the inspection satellite will re-attach (dock) to the main satellite.

Rendezvous and docking is categorised as follows: [14]

- Autonomous RVD (where the onboard computer is in full control), and
- Remotely piloted RVD (where the docking is done by a human operator from a remote location)

Autonomous RVD is preferred because:

- It does not rely on human capability. [14]
- It can be used for a wide variety of scenarios with various sized spacecraft. [14][29]
- Due to communication link constraints. A communication link to the ground is not always available and even if it is, it may not be practical due to limited bandwidth (if video must be used) or transmission delays that are always present. [4][6][14][25]
- It reduces the strain on limited human resources. (Ground crew and astronauts.) [29]
- RVD manoeuvres may result in the misalignment of antennas with the ground station or the target satellite may block the ground station resulting in a loss of communications during the docking process. [2]

Even though autonomous RVD has long been researched, it still poses many challenges. Multiple conditions and constraints have to be satisfied simultaneously. The onboard system must ensure that correct velocities and angular rates are maintained to ensure that the docking mechanisms remain precisely aligned. At the same time the safety of the spaceships must be ensured while also adhering to a time constraint. [6][21]

A prerequisite for autonomous docking is the ability to estimate the relative position, velocity and orientation between the chaser and target satellite. In this thesis, the satellite that will be docked at is referred to as the target satellite, while the satellite that will be docking is referred to as the chaser satellite.

A single camera (in other words monocular vision) is used to estimate the before mentioned quantities. It is considered to be the best overall sensor in terms of performance, mass and power consumption. [6] See chapter 7 for a more elaborate comparison of sensors.

1.2 Thesis Goals

The aim of this thesis is to demonstrate the concept of docking two satellites autonomously using monocular camera vision. This inherently implies:

- The research and design of a docking algorithm.
- Designing and implementing the camera sensor system.
- Demonstrating docking by creating a test platform.

A modified model hovercraft is used as a test platform as it moves relatively frictionless. This closely emulates the movement of a satellite in 2 dimensions.

1.3 Approach

Here follows the approach used in this thesis:

1. Literature search.
2. Design a docking algorithm based on the research.
3. Create the necessary sensor system based on the research. (Camera system, centroid detection, and light identification algorithm)
4. Design and build a test platform (hovercraft and base station) for emulation.
5. Create a simulation to test the docking concepts.
6. Implement the idea on the test platform.
7. Comment on the results.

1.4 Thesis Layout

Chapter 2 gives an overview of the whole docking sequence and introduces some necessary concepts. After this, the concept behind monocular camera vision is introduced and a literature overview of methods used to extract the target's pose (relative position and orientation of the target) from the images is given in chapter 3.

A light model is introduced in chapter 4 and is used to aid the identification and tracking of the target's centroids. Chapter 5 starts by first examining the movement of centroids in the image due to specific movements and continues by describing how the target's lights' centroids are uniquely identified in the image.

Then an overview of the target and chaser to be used in the emulation is given, from which the required hardware can be determined. This is then followed by a description of the hardware used and the calibration thereof in chapter 7. The camera and its calibration is discussed in detail in chapter 8.

Chapter 9 describes how Malan's EKF [20] is customised for this thesis' purposes. Next, the design and results of the inner loop controllers is given in chapter 10. In chapter 11 the system

Extended Kalman Filter (EKF) that is used to estimate the parameters required for docking is described. Chapter 12 describes the design of the outer loop controllers. The results of the simulation and emulation of the docking sequence is given in chapters 13 and 14, respectively.

This thesis' conclusion and recommendations are given in chapter 15.

1.5 Achievements

The following was achieved in this thesis:

1. The autonomous and reliable identification of markers was successfully demonstrated in both simulation and emulation.
2. The autonomous docking of a satellite pair using monocular vision was successfully demonstrated in both simulation and emulation.

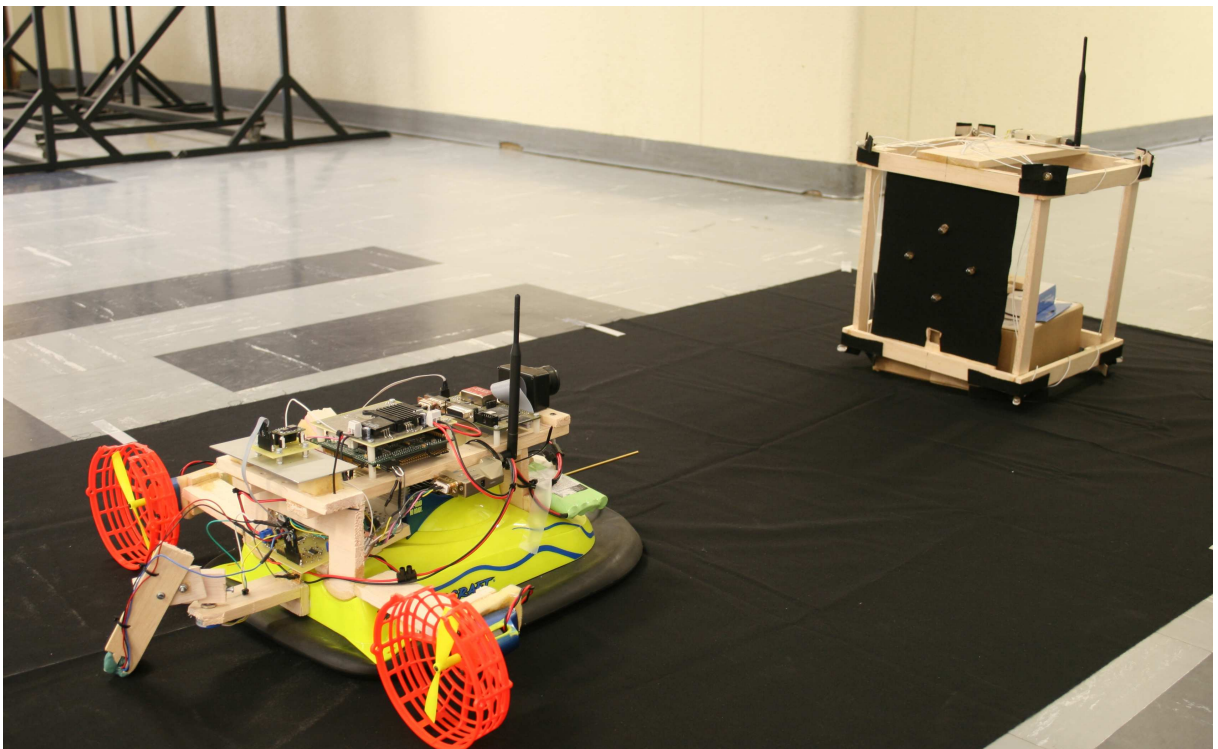


Figure 1.1: The emulation setup. Note that in this photo the black cloth is not fully tensioned.

Chapter 2

The Docking Sequence

The proposed docking sequence is presented in this chapter . Although most of the concepts will be presented in 2 dimensions, they apply directly to 3 dimensions. Typically, one should then only use a spherical coordinate system instead of a radial coordinate system. (In other words, only an extra angle must be considered.)

2.1 A Few Definitions

In this section the parameters and coordinate systems, that are required to explain the docking sequence, are introduced. Again it is stressed that for explanatory purposes the movement will be restricted to a 2 dimensional plane. Referring to figure 2.1, here are some definitions:

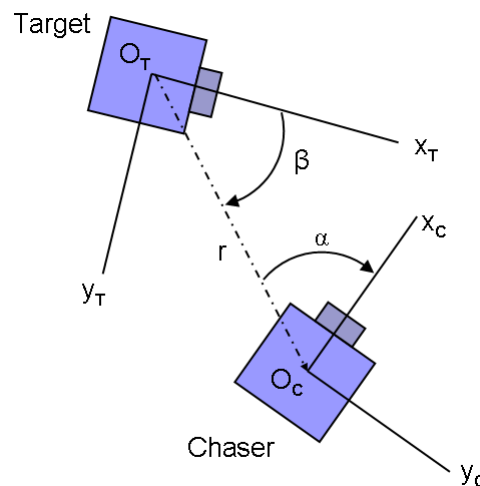


Figure 2.1: Parameter and axis definitions.

The chaser's body axis (x_C, y_C, z_C) is defined as follows: The z-axis normally points straight down (nadir) and goes through the chaser's centre of gravity. The x-axis coincides with the docking mechanism and the y-axis completes the right handed coordinate system. Name this coordinate system's origin O_C .

Similarly, the target's body axis (x_T, y_T, z_T) is defined as follows: The z-axis normally points straight down (nadir) and coincides with the target's axis of rotation. Again, the x-axis coin-

cides with the docking mechanism and the y-axis completes the right handed coordinate system. Name this coordinate system's origin O_T .

Let r be the length of the line, $\overline{O_C O_T}$, between the chaser's and target's body axes' origins. Define α as the angle between this line and the chaser's x-axis. Also, define β as the angle between this line and the target's x-axis.

Note that r and β forms a radial axis system whose origin is at O_T . There will be referred to this radial axis as the target's radial axis.

In this thesis, the camera is mounted in such a way that it points in the direction of the chaser's positive x-axis and its optical axis runs parallel to the chaser's x-axis. Therefore α is also the angle between the camera's optical axis and the line $\overline{O_C O_T}$.

The chaser's angular velocity around its z-axis is ω_{C_z} and the target's angular velocity around its z-axis is ω_{T_z} . Lastly, the angular velocity of the chaser around the target is ω_{C_T} .

As will be discussed later, the docking sequence consists out of three phases. For convenience sake, the three phases will be named:

1. Far Approach ($r > r_{FAR}$)
2. Close Approach ($r_{SAFE} < r < r_{FAR}$)
3. Final Approach ($r < r_{SAFE}$)

Define r_{FAR} as the distance between the satellites such that the target satellite is just visible to the chaser satellite's camera. Also, define r_{SAFE} as the radius of a sphere (from now on referred to as the "safety sphere") that encapsulates the whole target satellite along with its external components such as antennas and solar panels. (This safety sphere may not be entered by the chaser unless it is safe to do so.) Figure 2.2 illustrates this graphically.

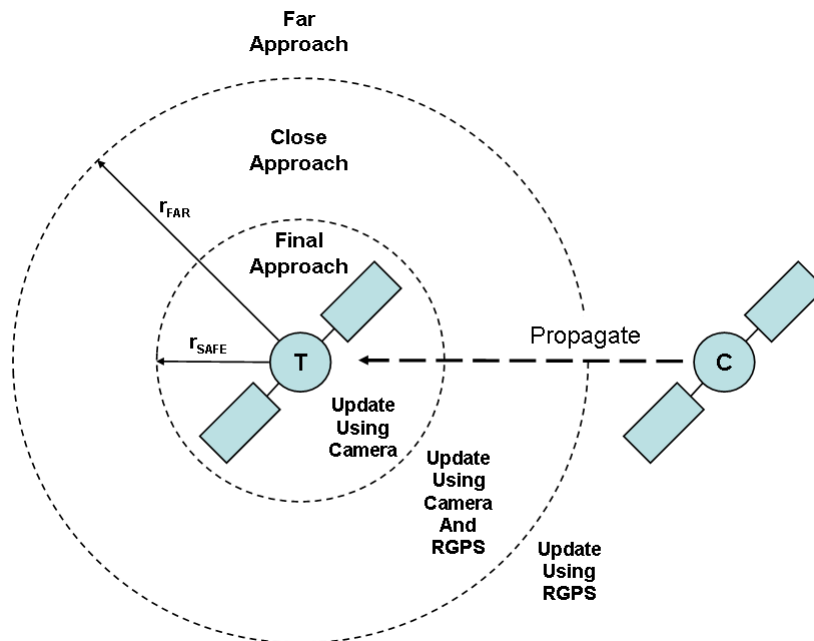


Figure 2.2: Phase definitions. The left satellite is the (T)arget and the right satellite is the (C)haser.

2.2 Overview of the Docking Sequence

A low impact docking sequence similar to the one proposed by Kawano *et al.* [14] will be used. The main advantage of a low impact docking sequence is that smaller, light weight docking mechanisms can be used instead of the more robust, heavier and larger mechanisms required by conventional impact docking. As this thesis is done with smaller satellites (such as nanosatellites) in mind, the smaller docking mechanisms are an attractive option. The price for this, on the other side, is a more complex control system that can control the relative velocities and angular rates more precisely.

A three stage approach is used as it provides a logical way to divide the docking sequence into subproblems. The current stage is a function of the relative distance between the target and chaser. Each stage can then have its own controller and estimator as necessary. [14][29]

Qureshi *et al.* [25] defines autonomous RVD as follow: "Autonomy entails that the onboard controller be capable of estimating and tracking the pose (position and orientation) of the target satellite and guiding the servicing spacecraft as it 1) approaches the satellite, 2) maneuver itself to get into docking position, and 3) docks with the satellite."

This fully describes the whole docking sequence, as well as the purpose of each phase, in a nutshell.

2.3 Detailed Description

Now a detailed description of the docking procedure can be given using figures 2.1 and 2.2, as well as the definitions of section 2.1.

2.3.1 Far approach

Initially, when the satellites are far apart, $r > r_{FAR}$, they will approach each other using relative GPS (RGPS). During this phase the primary goal of the controller should be to decrease the relative distance, r , between the satellites. The secondary goal would be to keep the chaser's camera pointing at the target by keeping α relatively small using the RGPS data. During the whole docking sequence, the target satellite is not allowed to make any sudden movement changes such as, for example, firing a thruster. However, during this first phase, this constraint can be relaxed as it is negligible due to the large distance between the satellites.

2.3.2 Close approach

During this phase, $r_{SAFE} < r < r_{FAR}$, the chaser satellite's camera sensor will be enabled and initialised. Therefore, from this phase onwards, the target satellite may not make any sudden movement changes.

For the cameras initialisation it is important to keep α relatively constant. As $\dot{\alpha} = \omega_{C_Z} - \omega_{C_T}$, this implies that $|\omega_{C_Z} - \omega_{C_T}|$ must be minimised. This is because small changes in α causes significant displacements of the target satellite in the image. Note that it is not necessary to keep the target satellite centred in the image, but to keep the target satellite stationary in the image. (As long as the target satellite is in the camera's field of view.)

It is also important that the difference between the angular velocity of the chaser around the target and the target's angular velocity, $|\omega_{Tz} - \omega_{Cz}|$, should be small during the camera sensor's initialisation phase. Otherwise the target's lights will move too fast in the image.

The current controller's goal should be to keep α relatively constant during the camera's initialisation process. While the camera is busy initialising, the distance between the satellites can slowly be decreased. After the camera was initialised, more aggressive manoeuvres are possible. This allows for angular velocity synchronisation. The chaser's current controller can now start to regulate α and β to zero so that eventually all the angular velocities are equal ($\omega_{Cz} = \omega_{Tz} = \omega_{Cz}$).

As the distance between the satellites decrease, the RGPS measurement's weight will be decreased while the camera measurement's weight will be increased until the much more accurate camera measurement has completely replaced the less accurate RGPS measurement.

2.3.3 Final approach

This last phase, $r < r_{SAFE}$, is the most critical phase of the whole docking sequence as it requires close proximity manoeuvres within the target's safety sphere. Therefore, the docking sequence will be aborted if anything appears suspect.

Before the chaser satellite may enter the target satellite's safety sphere, the following conditions must be met:

1. The docking mechanisms must be aligned, i.e. $\alpha = \beta = 0$.
2. The chaser and target satellites rotations must be synchronised, i.e. $\omega_{Cz} = \omega_{Tz} = \omega_{Cz}$.

If all goes well, the docking sequence ends with a low velocity impact (Typically 1 cm.s^{-1}).

To ensure the safety of the the satellites, the docking sequence will be aborted if either $|\alpha|$ or $|\beta|$ exceeds predetermined values ϵ_α and ϵ_β , respectively. Stated differently, the docking sequence will be aborted if $|\alpha| > \epsilon_\alpha$ or $|\beta| > \epsilon_\beta$. Note that both ϵ_α and ϵ_β are positive constants. Aborting causes the chaser satellite to retreat to a predefined distance, $r_{RETREAT}$, where $r_{RETREAT} > r_{SAFE}$.

The target's safety sphere and the constraint on $|\beta|$ can be represented graphically in terms of an "approach corridor" and a "keep out zone" as shown in figure 2.3.

2.4 Guidance System

Different actions have to be executed during each phase of the docking sequence. It is the guidance system's task to determine in what phase the docking sequence is in at any given time. From this it can decide what action has to be taken and can schedule the necessary controller, references and estimators accordingly.

In this thesis, a single Extended Kalman Filter (EKF) will be used to store the system's states. One of these states is the distance between the chaser and the target. This information is then used by the guidance system to perform the necessary actions. Figure 2.4 shows the guidance system, conceptually. The system's EKF is discussed in chapter 11.

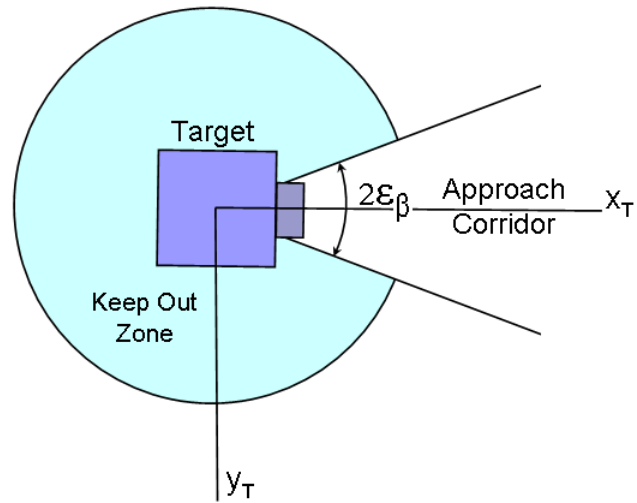


Figure 2.3: Target's approach corridor and keep out zone.

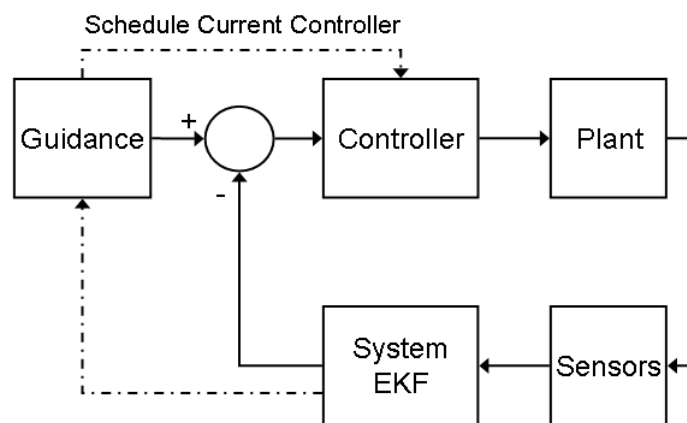


Figure 2.4: Conceptual representation of the guidance system (Adapted from [6]).

Chapter 3

Using a Monocular Camera

This chapter gives a description of what monocular camera vision entails, the methods used and the requirements.

3.1 Basic Principle

This section describes the basic principle of monocular camera vision. To do this, first consider stereo vision. Given two arbitrary points in space, a single camera can be used to determine the heading to each of these points. So far, the points' locations are not fixed as they can lie anywhere along these two vectors as illustrated in figure 3.1 (a). By using a second camera, the heading to each of these points can be acquired from a different reference point as seen in figure 3.1 (b). Now, using triangulation, the exact location of these points can be determined. This is the principal of stereo vision.

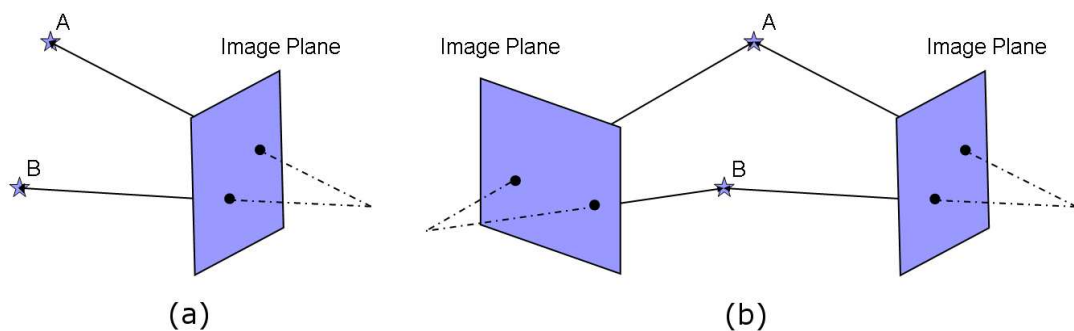


Figure 3.1: Stereo vision.

The basic principle of monocular vision is to utilise all the information at ones disposal. As docking will typically happen to a known satellite, with known dimensions, the distance between these points are known. Using this additional information, the locations of these two points can be determined. (Note that although this is the basic principle, several points are required for a unique solution.) Refer to figure 3.2. Then by observing a sequence of images, the relative velocities and angular rates can be estimated.

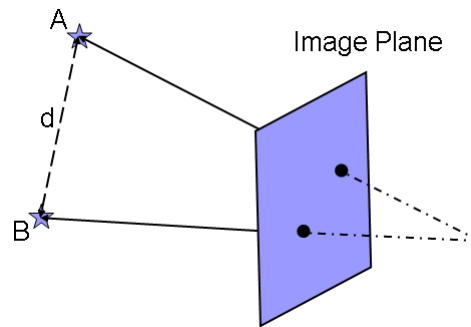


Figure 3.2: Monocular vision.

3.2 Monocular Vision versus Stereo Vision

Normally, stereo vision is associated with camera sensor systems. Monocular camera vision is preferred for satellite applications, i.e. using one camera instead of two implies half the cost, weight, power and volume. The smaller the satellite, the more significant these savings become.

3.3 Methods Considered

When monocular vision is used, several strategies are implemented to extract the required information from the successive photos. Some of them are:

- By using multi-coloured markers. Miller *et al.* [21] proposed the use of markers that are bright colours not normally found in an orbital setting. Using a colour camera these features are easily identifiable. This approach requires 3 markers that are not coplanar. As a gray scale camera is used in this thesis, this approach is not possible.
- Line models. Cropp *et al.* [2] extracts all the lines from a picture using the Hough transform. The identified lines are then compared to a line-based model of the target satellite using a heuristic approach. This method proved to be fairly reliable, but takes roughly 1 minute to process a single image on a Pentium 75 MHz. The method is therefore very processor intensive as one would expect from an algorithm that uses the Hough transform. The camera used in this thesis only has an 8051 microcontroller and therefore this approach is not feasible.
- Using the Scale Invariant Feature Transforms (SIFT) was explored. The basic principle of SIFT is that distinctive invariant features are extracted from an image. These features are scale and rotation invariant and can therefore be used to reliably match different views of an object. The process is described by D. Lowe. [17]

For this application the idea was to take a photo of each of the target satellite's 6 facets. These images would form the reference images whose features would be extracted and stored in a database. To determine the target satellite's pose, the following process will be followed:

1. Extract features from the current image.
2. Match these features to those in the database.
3. Use the results to determine the target's pose.

From Lowe's [17] description it can be seen that SIFT is extremely processor intensive and therefore not possible to be implemented on the 8051 microcontroller used.

- Another approach considered was to place unique patterns on each of the satellite's facets. These patterns would either be passive (such as markings) or active (such as light clusters). This will however not work as they can be unidentifiable or, even worse, be mistakenly identified when viewed from different angles.
- Lastly, light point sources can be placed on the target satellite. These appear in the image as roundish blobs of which the centroids can be determined. These centroids can then be processed to determine the target satellite's pose.

The determination of light centroids can fairly easily be accomplished on the 8051. This approach will therefore be used as it is the most compatible with the current hardware setup. For the more computationally expensive approaches, the hardware setup has to allow the images to be speedily delivered to a location where it can be accessed and processed by a capable processor.

When light point sources are used, there are two possible approaches:

- Relative angles only approach. As the name suggests, a camera is used to take angular measurements to markers on the target satellite. These measurements are then processed to determine the target's pose. This process is described in detail by Woffinden *et al.* [29]
- Using Malan's Extended Kalman Filter (EKF). In his thesis, Malan designed an EKF to estimate the pose of a target satellite directly from the centroids. [20]

Both of the above methods requires the ability to uniquely identify each centroid. The term "unique" implies two things: 1) One must know that the centroid belongs to the satellite and not to, for example, a star. 2) One must know exactly what light is represented by a specific centroid. This thesis will implement Malan's filter to test it and to use another, different approach to the methods already found in the literature.

3.4 Requirements

Malan's filter has to be provided with measurements in the form of 2D-3D point pairs. Markers (lights) will be placed on the target satellite. The position of these markers in the target's body axis are known exactly. These are the 3D points.

Then each marker has to be uniquely identified in the image. These centroids are the 2D points. Each centroid along with its corresponding position in the target's body axis forms a 2D-3D point pair. The unique identification and tracking of centroids is discussed in chapter 5. Malan's filter is tailored for the 2D emulation as discussed in chapter 9.

Chapter 4

Light Model

This chapter introduces a simple light model used to aid the tracking and identification of new centroids. The light model determines the position of the light centroids that would be seen by an ideal camera (with no distortion) observing the target satellite. Each light's radiation pattern is also taken into account.

4.1 Definitions

Figure 4.1 introduces the necessary variables used in this model. The following variables are used:

- $\mathbf{r}_{L_n}^{CAM}$ is the position of the light n in the camera axis.
- \mathbf{r}_{TGT}^{CAM} is the position of the target satellite axis' origin in the camera axis.
- \mathbf{i}, \mathbf{j} and \mathbf{k} are the unit vectors along the camera's x-, y- and z-axis, respectively.
- $\mathbf{r}_{L_n}^{TGT}$ is the position of the light n in the target satellite's axis.
- $\mathbf{e}_{L_n}^{TGT}$ is an unit vector giving the direction of the light radiation pattern's symmetry axis of the light n in the target satellite's axis.
- γ_n is the viewing angle of light n .
- $\mathbf{T}_{CAM \rightarrow TGT}$ is the Direction Cosine Matrix (DCM) that gives the target satellite axis orientation with respect to the camera axis.

For now, the camera axis ($x_{CAM}, y_{CAM}, z_{CAM}$) is defined as follows: The z-axis coincides with the camera's optical (principle) axis. The x- and y-axis completes the right handed coordinate system.

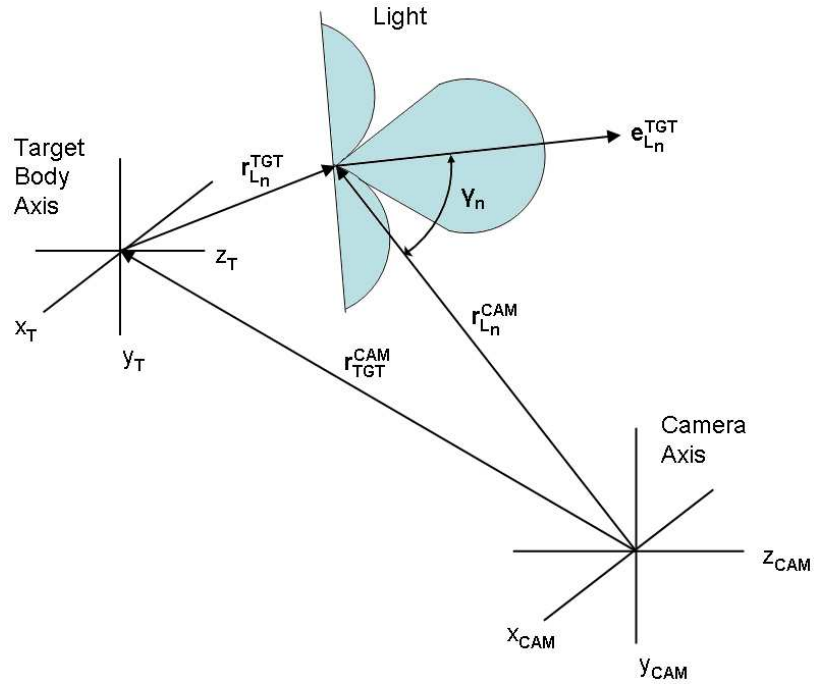


Figure 4.1: Light model.

4.2 Light Radiation Pattern

Each light has its own light radiation pattern. This light model assumes a symmetric light radiation pattern.

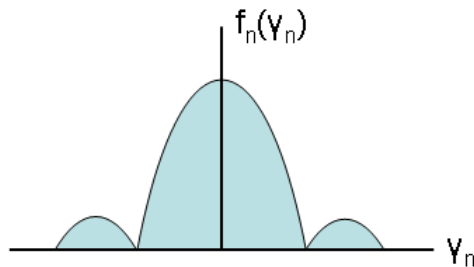


Figure 4.2: A light radiation pattern.

The light intensity for a specific viewing angle, γ_n , is given by:

$$I_{Light_n} = f_n(\gamma_n)$$

4.3 Pinhole Camera Model

Figure 4.3 shows the pinhole camera model for a projective camera with a focal length, f meters. The camera's z_{CAM} axis coincides with the camera's optical axis. (Also called the principle axis.) Where the optical axis intersects with the image plane ($z_{CAM} = f$) is known as the principle point $[u_0, v_0]^T$ where both u_0 and v_0 are in pixel units.

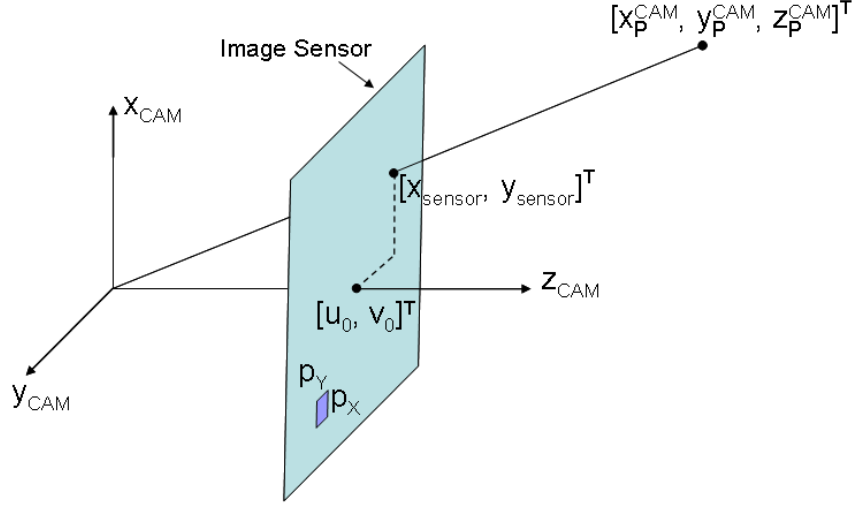


Figure 4.3: Pinhole camera model.

Given that the size of each pixel is p_X and p_Y meters along the camera's x - and y -axis, respectively, the projection $[x_{sensor}, y_{sensor}]^T$, in pixels, of an arbitrary point $[x_p^{CAM}, y_p^{CAM}, z_p^{CAM}]^T$ in the camera axis on the image plane is:

$$x_{sensor} = \left(\frac{x_p^{CAM}}{z_p^{CAM}} \right) \left(\frac{f}{p_X} \right) + u_0 \quad (4.3.1)$$

$$y_{sensor} = \left(\frac{y_p^{CAM}}{z_p^{CAM}} \right) \left(\frac{f}{p_Y} \right) + v_0 \quad (4.3.2)$$

4.4 The Light Model

4.4.1 Intuitive model

The position of a light n in the camera axis is given by:

$$\mathbf{r}_{L_n}^{CAM} = \mathbf{r}_{TGT}^{CAM} + (\mathbf{T}_{CAM \rightarrow TGT})^{-1} \mathbf{r}_{L_n}^{TGT}$$

Note that all the DCM matrices are orthonormal matrices so that their inverse is equal to their transpose, e.g. $(\mathbf{T}_{CAM \rightarrow TGT})^{-1} = (\mathbf{T}_{CAM \rightarrow TGT})^T$. From equations 4.3.1 and 4.3.2, the position of the centroid of light n in the image $[x_{sensor_n}, y_{sensor_n}]^T$, in pixels, is given by:

$$\begin{aligned} x_{sensor_n} &= \left(\frac{\text{proj}_{\mathbf{i}} \mathbf{r}_{L_n}^{CAM}}{\text{proj}_{\mathbf{k}} \mathbf{r}_{L_n}^{CAM}} \right) \left(\frac{f}{p_x} \right) + u_0 \\ &= \left(\frac{(\mathbf{r}_{TGT}^{CAM} + (\mathbf{T}_{CAM \rightarrow TGT})^{-1} \mathbf{r}_{L_n}^{TGT}) \cdot \mathbf{i}}{(\mathbf{r}_{TGT}^{CAM} + (\mathbf{T}_{CAM \rightarrow TGT})^{-1} \mathbf{r}_{L_n}^{TGT}) \cdot \mathbf{k}} \right) \left(\frac{f}{p_x} \right) + u_0 \end{aligned} \quad (4.4.1)$$

$$\begin{aligned}
y_{sensor_n} &= \left(\frac{\text{proj}_{\mathbf{j}} \mathbf{r}_{L_n}^{CAM}}{\text{proj}_{\mathbf{k}} \mathbf{r}_{L_n}^{CAM}} \right) \left(\frac{f}{p_y} \right) + v_0 \\
&= \left(\frac{(\mathbf{r}_{TGT}^{CAM} + (\mathbf{T}_{CAM \rightarrow TGT})^{-1} \mathbf{r}_{L_n}^{TGT}) \cdot \mathbf{j}}{(\mathbf{r}_{TGT}^{CAM} + (\mathbf{T}_{CAM \rightarrow TGT})^{-1} \mathbf{r}_{L_n}^{TGT}) \cdot \mathbf{k}} \right) \left(\frac{f}{p_y} \right) + v_0
\end{aligned} \tag{4.4.2}$$

From the definition of the dot product, the viewing angle of light n is:

$$\gamma_n = \arccos \left(\frac{(-\mathbf{r}_{L_n}^{CAM}) \cdot (\mathbf{T}_{CAM \rightarrow TGT})^{-1} \mathbf{e}_{L_n}^{TGT}}{\|(-\mathbf{r}_{L_n}^{CAM})\| \|(\mathbf{T}_{CAM \rightarrow TGT})^{-1} \mathbf{e}_{L_n}^{TGT}\|} \right) \tag{4.4.3}$$

$$= \arccos \left(\frac{(-\mathbf{r}_{L_n}^{CAM}) \cdot (\mathbf{T}_{CAM \rightarrow TGT})^{-1} \mathbf{e}_{L_n}^{TGT}}{\|\mathbf{r}_{L_n}^{CAM}\|} \right) \tag{4.4.4}$$

The simplification from 4.4.3 to 4.4.4 can be made as a DCM is a rotation matrix and therefore does not alter a vectors magnitude, and $\mathbf{e}_{L_n}^{TGT}$ is an unity vector.

One problem with the basic mathematical camera models is that it does not take into account if an object is in front or behind the camera. For example: According to equation 4.3.1, the only difference between an object located at +Z and -Z is the location of its projection in the image plane. This is of course not accurate as the object located at -Z is behind the camera and should not cause a projection onto the image plane. To circumvent this problem, one must manually test if an object (light) is in front of the camera or not. This is done by testing the z-component of $\mathbf{r}_{L_n}^{CAM}$. If it is larger than zero, the object is in front of the camera. Mathematically, for the light to be in front of the camera:

$$\mathbf{r}_{L_n}^{CAM} \cdot \mathbf{k} > 0$$

4.4.2 An improved model

Although the model presented in section 4.4.1 is very intuitive, it is undesirable. If there are N lights, then N light position vectors have to be rotated and translated, and N light intensity vectors have to rotated. A much better approach would be to rather rotate the camera axis and use the light position vectors and intensity vectors as is.

Starting with:

$$\mathbf{r}_{L_n}^{CAM} = \mathbf{r}_{TGT}^{CAM} + (\mathbf{T}_{CAM \rightarrow TGT})^{-1} \mathbf{r}_{L_n}^{TGT}$$

Multiply, from the left, with $\mathbf{T}_{CAM \rightarrow TGT}$ and call the result $\hat{\mathbf{r}}_{L_n}$:

$$\begin{aligned}
(\mathbf{T}_{CAM \rightarrow TGT}) \mathbf{r}_{L_n}^{CAM} &= (\mathbf{T}_{CAM \rightarrow TGT}) \mathbf{r}_{TGT}^{CAM} + \mathbf{r}_{L_n}^{TGT} \\
&= \hat{\mathbf{r}}_{L_n}
\end{aligned} \tag{4.4.5}$$

The rotated camera axis becomes:

$$\begin{aligned}(\mathbf{T}_{CAM \rightarrow TGT})\mathbf{i} &= \hat{\mathbf{i}} \\(\mathbf{T}_{CAM \rightarrow TGT})\mathbf{j} &= \hat{\mathbf{j}} \\(\mathbf{T}_{CAM \rightarrow TGT})\mathbf{k} &= \hat{\mathbf{k}}\end{aligned}$$

Notice that as $\mathbf{i} = [1, 0, 0]^T$, $\mathbf{j} = [0, 1, 0]^T$ and $\mathbf{k} = [0, 0, 1]^T$, the vectors $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ are just the first, second and third column of $\mathbf{T}_{CAM \rightarrow TGT}$, respectively.

Then, the position of the centroid $[x_{sensor_n}, y_{sensor_n}]^T$ of light n is given by:

$$\begin{aligned}x_{sensor_n} &= \left(\frac{\text{proj}_{\hat{\mathbf{i}}} \hat{\mathbf{r}}_{L_n}}{\text{proj}_{\hat{\mathbf{k}}} \hat{\mathbf{r}}_{L_n}} \right) \left(\frac{f}{p_x} \right) + u_0 \\ &= \left(\frac{\hat{\mathbf{r}}_{L_n} \cdot \hat{\mathbf{i}}}{\hat{\mathbf{r}}_{L_n} \cdot \hat{\mathbf{k}}} \right) \left(\frac{f}{p_x} \right) + u_0\end{aligned}\tag{4.4.6}$$

$$\begin{aligned}y_{sensor_n} &= \left(\frac{\text{proj}_{\hat{\mathbf{j}}} \hat{\mathbf{r}}_{L_n}}{\text{proj}_{\hat{\mathbf{k}}} \hat{\mathbf{r}}_{L_n}} \right) \left(\frac{f}{p_y} \right) + v_0 \\ &= \left(\frac{\hat{\mathbf{r}}_{L_n} \cdot \hat{\mathbf{j}}}{\hat{\mathbf{r}}_{L_n} \cdot \hat{\mathbf{k}}} \right) \left(\frac{f}{p_y} \right) + v_0\end{aligned}\tag{4.4.7}$$

In order to derive the equation for γ_n use the identity that $\mathbf{T}\mathbf{a} \cdot \mathbf{T}\mathbf{b} = \mathbf{a} \cdot \mathbf{b}$ if \mathbf{a} and \mathbf{b} are vectors and \mathbf{T} is a rotation matrix. This is true as a rotation matrix does not alter a vector's magnitude. Also, if both vectors are rotated the same way, the angle between them remains the same. The viewing angle of light n is now given by:

$$\gamma_n = \arccos \left(\frac{-(\mathbf{T}_{CAM \rightarrow TGT})\mathbf{r}_{L_n}^{CAM} \cdot \mathbf{e}_{L_n}^{TGT}}{\|-(\mathbf{T}_{CAM \rightarrow TGT})\mathbf{r}_{L_n}^{CAM}\| \|\mathbf{e}_{L_n}^{TGT}\|} \right)\tag{4.4.8}$$

$$= \arccos \left(\frac{-\hat{\mathbf{r}}_{L_n} \cdot \mathbf{e}_{L_n}^{TGT}}{\|\hat{\mathbf{r}}_{L_n}\|} \right)\tag{4.4.9}$$

The simplification from 4.4.8 to 4.4.9 is made using equation 4.4.5.

Lastly, to test if the object is in front of the camera: The light is in front of the camera if

$$\hat{\mathbf{r}}_{L_n} \cdot \hat{\mathbf{k}} > 0\tag{4.4.10}$$

4.4.3 Using the improved model

The procedure to use the improved model is as follows:

1. Calculate $(\mathbf{T}_{CAM \rightarrow TGT})\mathbf{r}_{TGT}^{CAM}$.
2. Calculate $\hat{\mathbf{i}}, \hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$.
3. For each light n:
 - a) Calculate $\hat{\mathbf{r}}_{L_n}$ using equation 4.4.5.
 - b) If $\hat{\mathbf{r}}_{L_n} \cdot \hat{\mathbf{k}} > 0$:
 - i. Calculate $[x_{sensor_n}, y_{sensor_n}]^T$ using equations 4.4.6 and 4.4.7.
 - ii. Determine γ_n using equation 4.4.9, and from this and the light's radiation pattern the centroid's intensity.
 - iii. Use/Store this light's centroid data.

From the preceding procedure, one can see why the improved model is better than the intuitive one. Here, only 4 rotations have to be calculated of which 3 are only the columns of a rotation matrix. Therefore this model is much less processor intensive, especially when a large number of lights are used.

4.5 Conclusion

The light model introduced in this chapter is used to track and identify new centroids as will be explained in chapter 5. It also acts as a virtual camera in the simulation discussed in chapter 13.

Chapter 5

Identification and Tracking of Light Centroids

In this chapter the centroid identification and tracking unit is introduced. The algorithm used to determine the centroids is also discussed.

5.1 Overview

The purpose of the centroid identification and tracking unit is to uniquely identify and track the light features (markers) on the target satellite in the presence of noise sources such as stars, especially the sun, and reflections off the target satellite's surface. It therefore forms the connection between the raw centroids received from the camera unit and Malan's filter (Refer to figure 6.1). In this regard, this unit must be accurate and reliable enough to handle centroids that appear and disappear. The light model described in chapter 4 is used to aid this process.

This centroid identification and tracking process can be divided into three subprocesses as illustrated in figure 5.1:

- **Initial search.** As soon as the target satellite is close enough, the centroids must be systematically identified. This process can be slow as it is typically only performed once.
- **Tracking.** After a centroid was identified, its position has to be tracked in successive images. This process should be as fast as possible as it is frequently executed.
- **Detect changes.** Due to the target's and chaser's relative movement (rotation and translation) some features will appear and disappear from the camera's field of view. Therefore it is necessary to determine which centroids fell away and to be able to identify the new centroids that became visible. This process is also executed frequently and should also be as fast as possible.

In this thesis, a communications link is also available for utilisation. (i.e. the target is cooperative.) Of course, if the docking sequence can be executed without using the communications link (i.e. a uncooperative target), the algorithm will be much more robust as it is not dependent on an intersatellite communications link. But this approach is inherently more complex.

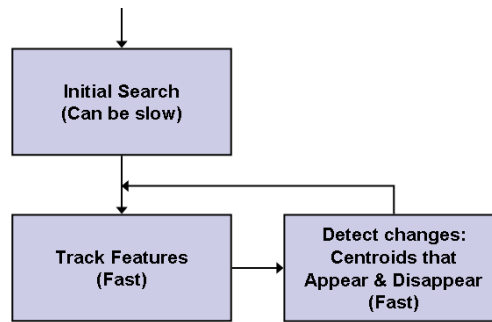


Figure 5.1: Breakdown of the identification and tracking process.

5.2 Movement of the Target Satellite's Centroids and the Star's Centroids in the Image

Before one can design an algorithm to track centroids in successive images, one must first investigate the movement of centroids in successive images.

In sections 5.2.1 to 5.2.5 the movement of the noise source's (e.g. stars) centroids and the movement of the target satellite's lights' centroids in the image, as the result of specific actions, are examined.

5.2.1 Movement of stars' centroids due to the rotation and translation of the satellites around one another and the earth

The movement of stars' centroids in successive images, due to the rotation and translation of the satellites around one another and the earth, is negligible.

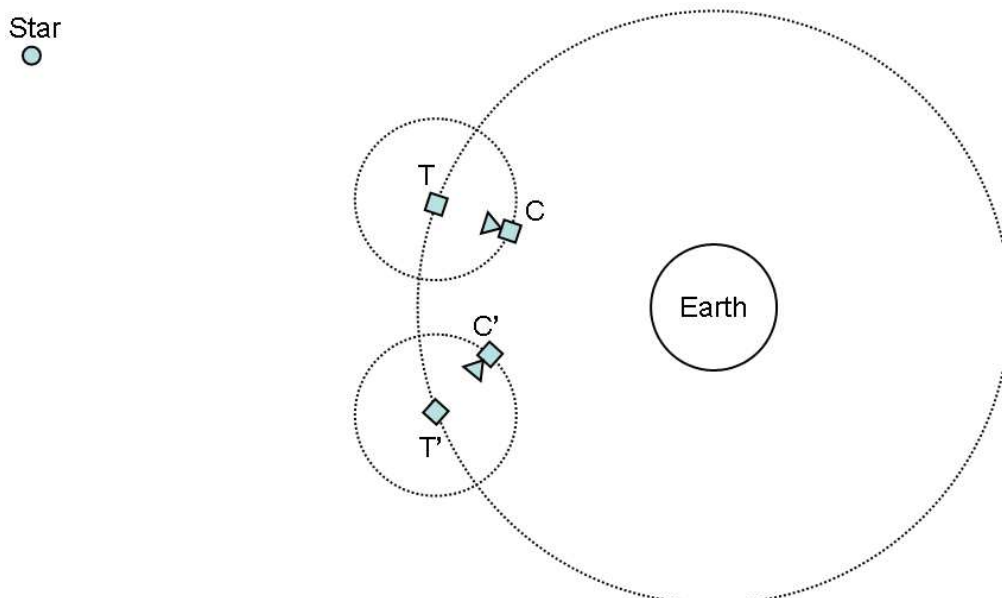


Figure 5.2: Movement of the chaser and target around each other and the earth.

Figure 5.2 illustrates how the target satellite orbits around the earth from T to T' while the chaser simultaneously rotates around the target from C to C' . If the chaser keeps the target centred in the image, then only the star's centroid will move in the image. As result of the movement depicted in figure 5.2, the star's centroid will move from its old position (circle) to its new position (triangle) in the image as shown in figure 5.3.

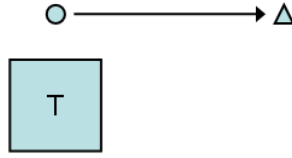


Figure 5.3: The star's centroid move from its old position (circle) to its new position (triangle), while the target satellite (square) remains stationary in the image.

Given that the time between two successive images is ΔT , it is shown in section B.1 that the displacement of the star's centroid in the image is given by:

$$d_{star} = \left(\frac{f}{p} \right) \tan(2\omega_{C_T} \Delta T) \text{ pixels}$$

In order to estimate d_{star} , consider the following: The rate at which the chaser and target satellites orbit around each other is equal to the rate at which the satellites orbit around the earth. As a typical orbital period for low earth orbital satellites is about 90 minutes, the rate at which the satellites orbit around each other, ω_{C_T} , is given by

$$\begin{aligned} \omega_{C_T} &= \frac{2\pi}{(90)(60)} \\ &= 1.164 \times 10^{-3} \text{ rad.s}^{-1} \end{aligned}$$

The lens' datasheets gives $f = 4 \times 10^{-3}$ m and $p = 6 \times 10^{-6}$ m. Lastly, the time between two successive photos, ΔT , will be less than 1 second. Therefore, the resulting distance moved by a star in the image would be less than 1.552 pixels.

5.2.2 Movement of stars' centroids due to the rotation of the chaser satellite around its own axis

The stars' centroids will move around significantly due to the rotation of the chaser satellite, or more specifically the camera, around the chasers' own axis. (For a rigid body these rotations are the same.)

When the chaser rotates through an angle of θ_{C_z} around its own z-axis (figures 5.4a and 5.4b), the star's centroid moves from its old position (circle) to its new position (triangle) as shown in figure 5.4c.

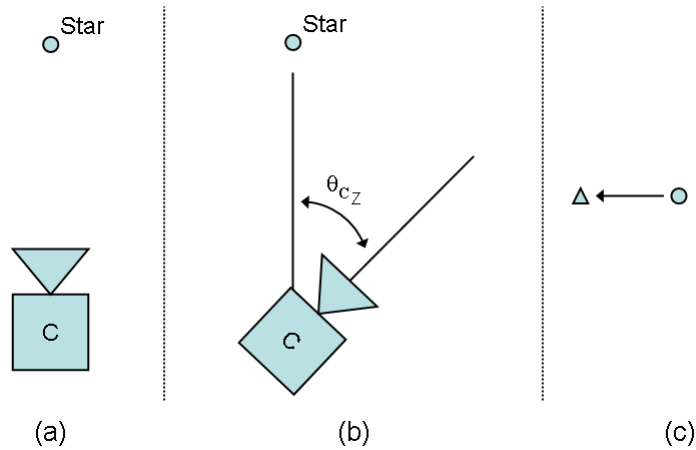


Figure 5.4: Movement of star's centroid due to the rotation of the chaser satellite (C) around its own axis.

It can be shown (refer to section B.2) that the displacement of a star's centroid in the image after the chaser rotated through θ_{Cz} around its z-axis is given by:

$$d_{star} = \left(\frac{f}{p} \right) \tan(\theta_{Cz}) \text{ pixels}$$

From this equation it can be seen that the displacement of the stars' centroids in the image is very sensitive to θ_{Cz} .

5.2.3 Movement of the target's lights' centroids due to the relative rotation of the satellites

If there is little difference in the angular rate of the target around its own axis, the chaser's angular rate around its own axis, and the angular rate of the chaser around the target ($\omega_{Tz} \approx \omega_{Cz} \approx \omega_{C_T}$), then the target's lights' centroids will keep their relative position and orientation in successive images as illustrated in figure 5.5.

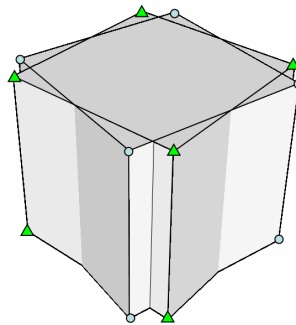


Figure 5.5: Movement of the target's lights' centroids due to the relative rotation of the satellites.

In figure 5.5 the light centroids have moved from their old positions (circles) to their new positions (triangles) during two successive photos. Note that the new positions are very close to the old positions if $\omega_{Tz} \approx \omega_{Cz} \approx \omega_{C_T}$.

5.2.4 Movement of the target's lights' centroids due to the rotation of the chaser around its own axis

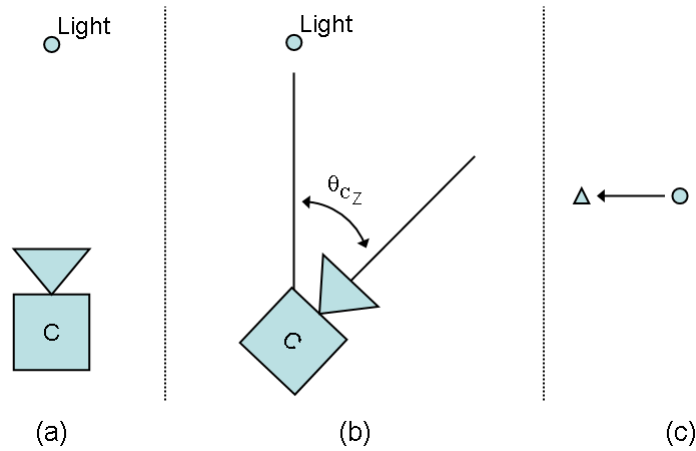


Figure 5.6: Movement of the target's light's centroid due to the rotation of the chaser around its own axis.

If the chaser satellite rotates about its own axis, then the target's lights' centroids will be displaced significantly in the image as illustrated in figure 5.6. The displacement of a light centroid in the image is given by:

$$d_{light} = \left(\frac{f}{p} \right) \tan(\theta_{CZ}) \text{ pixels}$$

Where θ_{CZ} is the angle through which the chaser satellite turned between the two successive photos. From this equation it can be seen that the displacement of the target's centroids in the image is very sensitive to θ_{CZ} .

5.2.5 Movement of the target's lights' centroids due to the relative translation between the satellites

If the chaser moves closer to the target while keeping the target centred in the image (as illustrated in figure 5.7(a)), then the target's lights' centroids in the image will move away from each other. This is illustrated in figure 5.7(b) where the centroids move from their old positions (circles) to their new positions (triangles).

From figure 5.7(b) one can see that this movement gives the same effect as when one would have zoomed in. Define the zoom factor, Z_F , as the ratio between the distances (in pixels) between the new centroids and the distances (in pixels) between the old centroids. It is shown in section B.3 that:

$$\begin{aligned} Z_F &= \frac{Z}{Z - \Delta Z} \\ &= \frac{1}{1 - \frac{\Delta Z}{Z}} \end{aligned} \quad (5.2.1)$$

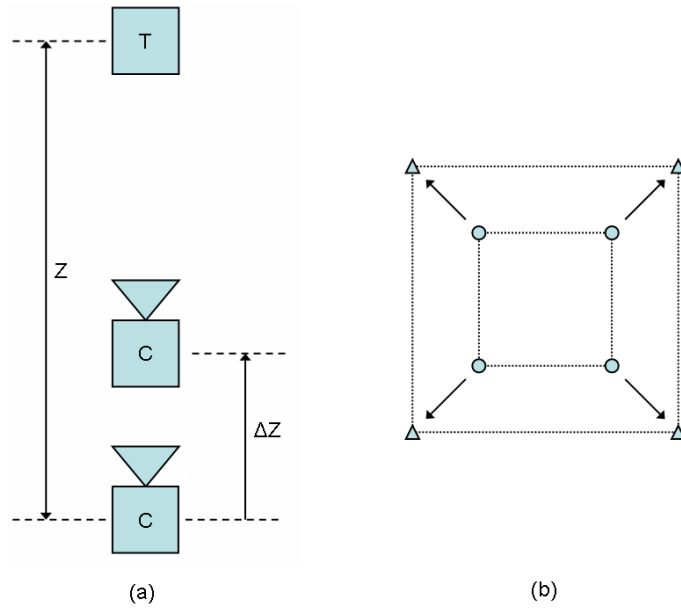


Figure 5.7: As the chaser moves closer to the target (a), the target’s lights’ centroids move away from each other in the image (b).

Where ΔZ is the distance travelled between two successive images and Z is the original distance between the satellites as shown in figure 5.7(a). From equation 5.2.1 it can be seen that if the distance travelled between two successive images is small in comparison to the distance between the satellites, then the zoom factor, Z_F , will be close to 1. This implies that for slow movements the the centroids will move very slowly in the image as they remain in almost the same position between successive images.

Now consider the relative translation along another axis as shown in figure 5.8(a). Translation along the other axes causes the target’s lights’ centroids to translate in the image. Note that because the chaser orbits around the target satellite, the translation shown in figure 5.8(a) will be the norm. Closer investigation shows that the translation of the centroids in the image is due to a pointing error that occurs when ω_{C_z} differs to much from ω_{C_T} . This can be seen by comparing figure 5.8(b) with figure 5.6.

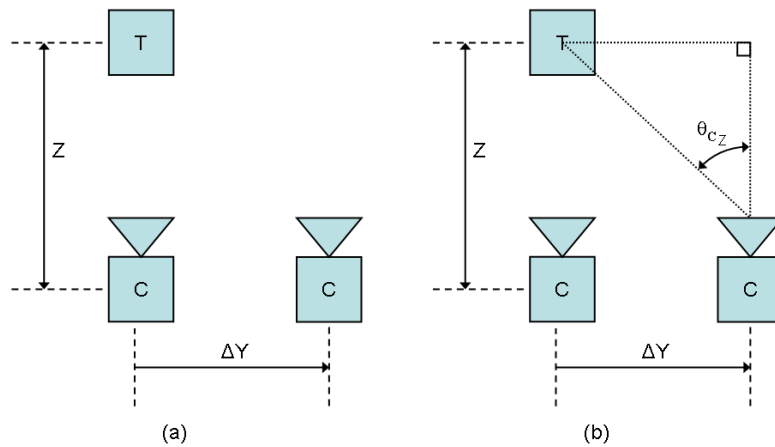


Figure 5.8: Relative translation of the chaser along another axis.

5.2.6 Conclusion

In this section the movement of stars' and the target's lights' centroids in the image due to certain movements were investigated to get a better understanding of what is required by the centroid identification and tracking unit. From this section it can be seen that most of the centroids move very slowly in the image due to the slow movements of the satellites. It was also seen that the movement of the centroids in the image is very sensitive to rotations of the chaser around its own axis.

In this application, the slow movement of the satellites is the characteristic that is exploited in the design of a matching algorithm. By ensuring that the rate of rotation of the chaser around its own axes is also slow, the centroids will move very slowly in the image so that the position of the centroids in the new image will be close to their positions in the older image.

5.3 The Matching Algorithm

A matching algorithm was designed to match centroids in successive images, i.e. determine the new position of each centroid from the old image in the new image. This is done by evaluating the distances between *all* of the old centroids and *all* the new centroids. Matching is then done by pairing up the old and new centroids that are closest to each other as it is known from section 5.2.6 that the centroids move very slowly in the image.

As an example consider the simplified problem shown in figure 5.9. In this example there are 4 centroids: 2 old centroids (circles) and 2 new centroids (triangles), where the old centroids 1 and 2 corresponds to the new centroids 1' and 2', respectively. Starting at centroid 1, the distances (d_{11} and d_{12}) to each new centroid (1' and 2') are determined. Next, the distances (d_{21} and d_{22}) from the old centroid 2 to each new centroid (1' and 2') are determined. Then all the distances are evaluated to determine the centroid pairs. As d_{11} is the shortest distance between the old centroid 1 and *any* other new centroid, the old centroid 1 is paired with the new centroid 1'. Similarly, as the distance d_{22} is the shortest distance between old centroid 2 and *any* other new centroid, old centroid 2 is paired with the new centroid 2'. Notice that the matching algorithm therefore tracks centroids in successive images.

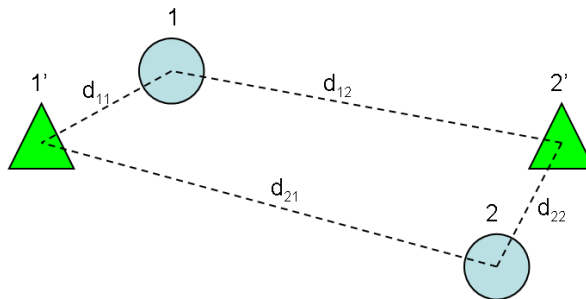


Figure 5.9: Matching old and new centroids.

This matching algorithm was implemented as a function in the C programming language. Apart from matching the old and new centroids, the function also reports which centroids disappeared (old centroids that had no match) and appeared (new centroids with no match). The

function also returns the number of centroids that changed in a variable called *change_amount*. This variable has a value $+N$ if there were N centroids that appeared, 0 if there was no change and $-N$ if there were N centroids that disappeared. All this information is used by the light identification and tracking unit as will be explained later.

It is stressed that this matching algorithm relies on the fact that the relative movement between the satellites is slow. If the relative movement becomes too aggressive, this matching algorithm will fail.

5.4 The Identification and Tracking Unit

In this section the identification and tracking process is described. Note that the chaser uses the wireless communications link to specify to the target which lights must be switched on or off. Refer to appendix A for the block diagrams.

5.4.1 Initial search

When the target satellite first becomes visible to the chaser, the target's lights' centroids has to be uniquely identified. But as the satellites are still quite far apart, the mass¹ of a centroid of a star and that of the target's light is similar. Under these circumstances it is extremely difficult to distinguish between the noise centroids, e.g. from the stars, and the target satellite's centroids.

One approach to distinguish between a light and a star would be to switch a light on and off multiple times to verify that the centroid is truly a light from the target satellite as the probability of a star repeatedly appearing and disappearing is zero. Taking into account that the pictures are taken and processed at a rate of less than 2 Hz, using multiple pictures for the detection of a single light increases the initialisation time drastically.

A much better approach is to first get the satellites as close together as possible by means of the other available sensors. As the distance between the satellites decreases, the mass of the target's lights' centroids increases. All the stars remain fixed at infinity so that their mass remains unchanged. Using the before mentioned method, the chaser satellite moves closer to the target satellite until the mass of the target satellite's lights' centroids are much larger than those of the stars. (Note that although the sun also remains at a relatively fixed distance, its centroid is fairly large. Fortunately, there is only one sun and its centroid can be tracked and compensated for.) Now the centroids can also be filtered by mass, so that the resulting centroids were filtered on three different ways:

1. Optically using, for example, an infra red filter.
2. By intensity. The camera unit thresholds the image before searching for centroids so that only centroids brighter than a certain threshold value are used.
3. By mass. Only centroids with a mass larger than a certain threshold value are accepted.

Now the initial light identification process can commence. This initialisation process is described here and the corresponding block diagram is shown in figure A.5 on page 154.

¹A centroid's mass is defined as the number of pixels that the star or target satellite's light consists of in the image.

The initialisation is done as follows:

1. All the target's lights are switched off before the first photo is taken. The resulting, filtered centroids are therefore from noise sources and will automatically be tracked by the matching algorithm introduced in section 5.3.
2. Next, the target's primary lights are systematically added. For each primary light the following procedure is followed:
 - a) Switch on the light.
 - b) Take a photo and search the whole image for centroids.
 - c) Filter the centroids by mass.
 - d) Match the centroids using the matching algorithm.
 - e) Evaluate the value that the variable *change_amount* returned by the matching algorithm:
 - If the number of changed centroids are zero, then the light is not visible to the camera. The light is then switched off to prevent a future, false trigger.
 - If the number of changed centroids is exactly one, then the centroid is identified as being the current light.
 - For any other number of changed centroids, the initialisation sequence is restarted as it is not possible to uniquely identify the centroid belonging to the current light under test.

3. After all the primary lights were tested, the number of identified centroids are counted.

If enough centroids were obtained, then the initialisation sequence is complete. It is now said that the identification and tracking unit has obtained lock. Based on which centroids were identified, Malan's EKF and the system EKF are initialised using heuristics. The identified centroids can now be sent to Malan's filter.

If the number of identified centroid are not enough, the initialisation process is restarted.

Malan [20] states in his thesis that his filter requires at least 4 centroids to work accurately. Therefore the minimum number of centroids that has to be identified was chosen as 4.

Note that the target satellite contains primary and secondary lights. The primary lights are used for long and medium distances. The secondary lights are enabled at close range and are used when, because of the close proximity, the primary lights can move out of the cameras field of view.

5.4.2 Tracking and detecting changes

After the initial light identification process is completed, the identified centroids have to be tracked in successive images. Due to the relative movement between the satellites there also

has to be tested regularly if any new lights have become visible and if any of the currently identified lights has disappeared. To track the centroids and to determine which centroids has disappeared is trivial as it is automatically done by the tracking algorithm from section 5.3.

As there is made use of the light model of chapter 4 to aid the identification of new light centroids, one has to wait for the system EKF to first converge as the light model uses the system EKF's state vector.

While the system EKF has not converged, the identified centroids are only tracked using the matching algorithm as described here (Refer to figure A.6 on page 155 for the block diagram):

1. Take a photo and search the whole image for centroids.
2. Filter the centroids by mass.
3. Match the centroids using the matching algorithm.
4. Evaluate the value that the variable *change_amount* returned by the matching algorithm:
 - If there is no change, the updated centroids are sent to Malan's filter.
 - If there are any new centroids, then they belong to noise sources. (Remember that all the unidentified lights are switched off.) Note that these centroids will now also be tracked by the matching algorithm. The updated light centroids are sent to Malan's filter.
 - In the case where centroids disappeared, it first has to be checked if the disappeared centroids belong to the target satellite's lights.
 - a) If the disappeared centroids do not belong to the target satellite's lights, the updated light centroids are sent to Malan's filter.
 - b) Else, the target's lights that correspond to the centroids that disappeared are switched off. The number of remaining light centroids are counted and if there are not enough light centroids remaining, lock is lost so that the initialisation process has to be redone. If there remain enough light centroids, they are sent to Malan's filter.

After the system EKF has converged, the light model from chapter 4 can be used to aid the tracking and identification of new centroids as described next (Refer to figure A.7 on page 156 for the block diagram):

1. Determine the model's centroids by using the light model and the system EKF's state vector. These centroids gives the position were centroids can be expected.
2. Set the location of the Windows Of Interest (WOI) to the model's centroid locations so that only these subsections of the image are searched for centroids. By only searching these small areas, the image is not only searched faster but the searching process also becomes more robust to noise sources.
3. Compare the model's centroids with the currently identified centroids. If the model returned centroids of lights that are not currently identified, then there is a possibility that these lights might be visible.

4. If there is a possibility that some of the unidentified lights may be visible, then they are systematically tested and added if found. This process is very similar to the initial search as will be shown:
 - a) Select one of the possible lights and switch it on.
 - b) Take a photo and search the WOIs for centroids.
 - c) Filter the centroids by mass.
 - d) Match the centroids using the matching algorithm.
 - e) Evaluate the value that the variable *change_amount* returned by the matching algorithm:
 - If the number of changed centroids is exactly one, then the light is identified.
 - If there are no new centroids or more than one new centroid, then the light under test is switched off as it cannot be uniquely identified.
 - If some centroids disappeared, the light under test is switched off. Then it is tested if the centroids that disappeared belongs to the target satellite's lights and if so, the corresponding lights are switched off. Again, the number of identified lights are counted and if there are not enough, lock is lost. The initialisation process then has to be redone. Else, if there is enough centroids left, the updated light centroids are sent to Malan's filter.

If there are no new possible lights, then the same tracking procedure is followed as when system EKF was not converged.

There are two advantages of using the light model to complement the centroid identification and tracking unit. The first is that it allows one to predict which lights will become visible. Secondly, it predicts where all the centroids will be, so that only the small areas in these locations (WOIs) have to be searched increasing the search speed and robustness of the search process. Therefore, the centroids resulting from this search process were filtered in 4 ways:

- Optically
- By intensity.
- By mass.
- By position using the WOIs.

For the purposes of detecting changes it is assumed that the target satellite's light centroids appear and disappear gradually as shown in figure 5.10. If it would happen that centroids appear and disappear in the same image, the matching algorithm will not detect it and would wrongly pair up the centroids from the previous and current image. Therefore, the design of the lights (their position and radiation pattern) must enforce this assumption.

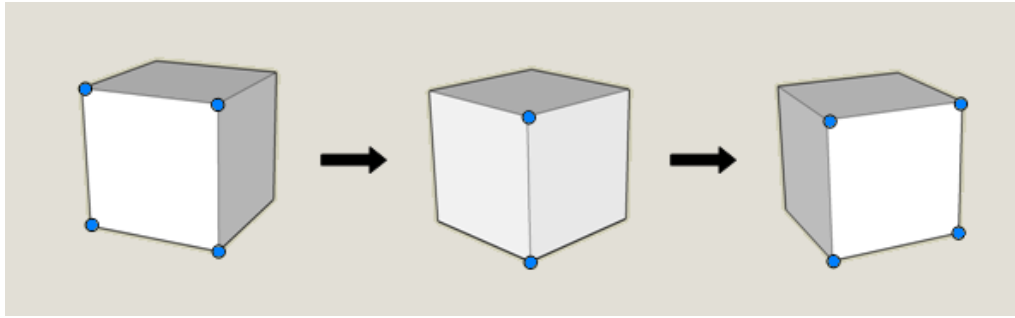


Figure 5.10: The gradual appearance and disappearance of centroids.

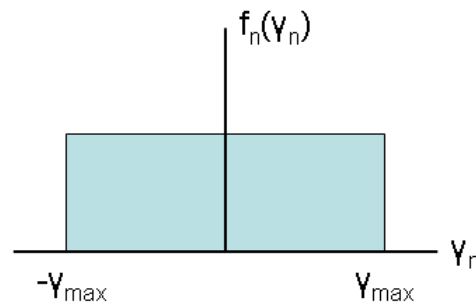


Figure 5.11: The light radiation pattern used in this thesis.

5.4.3 More on using the light model

From section 5.4.2 it can be seen that the light model is only used to predict if a light might be visible. Therefore, the simplified light radiation pattern shown in figure 5.11 is used.

It should be noted that the mathematical model must be used with caution as it is only a simplified model of the lights.

The main problem that can arise, is when the modelled light radiation pattern is smaller than the actual light's radiation pattern. (i.e. The actual light is visible from a wider angle than the model's light.) In this case the model's centroids will disappear before the actual light's centroids disappear. When this happens, no WOIs will be created for those lights and those centroids will be missed. This in turn will decrease the measurement's quality.

To circumvent this problem, one must ensure that the modelled light radiation pattern is always wider than the actual light radiation pattern. This will not cause any problems as the model's centroids are only used to specify the locations of the WOIs to be searched for centroids.

5.5 Searching for Centroids in the Image

When the image is search for centroids, the image is sub sampled to increase the search process' speed. It was determined empirically that the mass of the smallest centroid is 22 pixels. This corresponds to a circular light spot with a diameter of 5 pixels. Therefore, only the pixels in every 5th column of every 5th row is tested if its intensity is above the intensity threshold. In doing so, the search process is 25 times faster than searching each pixel. Then a centroid

determination algorithm is executed at the location of each pixel who's intensity has exceeded the threshold.

5.6 Rapid Centroid Determination Algorithm

One approach to determine the centroid of a light spot in the image is to use the region growing method as described by Jacobs [13]. One drawback of this method is that it results in a large amount of pixels being read 4 times. If the cost of this algorithm, C_{rg} , is defined as the number of pixels read, then it can be shown that:

$$C_{rg} = 4N + 1$$

where N is the number of pixels in the light spot.

Taking into account the limited processing capabilities of the camera's microcontroller and the fact that the mass² of the target satellite's marker can exceed 300 pixels at close proximity, it was decided to design a faster centroid determination algorithm. Apart from taking the image memory structure into account, the main feature of this new algorithm is that it only reads each pixel once. Refer to appendix C for more detail regarding this algorithm.

5.6.1 Comparing the faster centroid determination algorithm

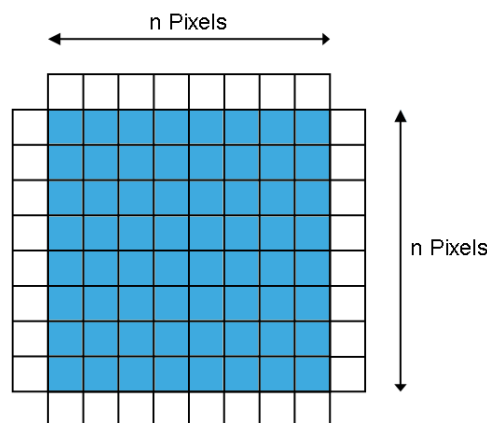


Figure 5.12: A $n \times n$ square (shaded) with bordering pixels (clear).

It is slightly more difficult to write down a general equation that describes the cost of the new algorithm, C_{new} . In words the cost is equal to the light spot's mass plus the number of pixels at the light spot's border. To simplify the comparison a square shaped light spot will be considered. Referring to figure 5.12, it can be shown that for a square having sides of length n pixels:

$$C_{new} = n^2 + 4n$$

²The number of pixels in the marker or light spot.

While:

$$C_{rg} = 4n^2 + 1$$

To show how much faster the new algorithm is, divide C_{rg} by C_{new} :

$$\begin{aligned} \eta &= \frac{C_{rg}}{C_{new}} \\ &= \frac{4n^2 + 1}{n^2 + 4n} \\ &= \frac{4 + \frac{1}{n^2}}{1 + \frac{4}{n}} \end{aligned} \quad (5.6.1)$$

From equation 5.6.1 and figure 5.13 it can be seen that for light spots with a small mass, there is little difference in performance. As the mass of a light spot increases, the faster algorithm becomes superior in terms of speed and is up to 4 times faster (if $n \rightarrow \infty$) than the region growing algorithm.

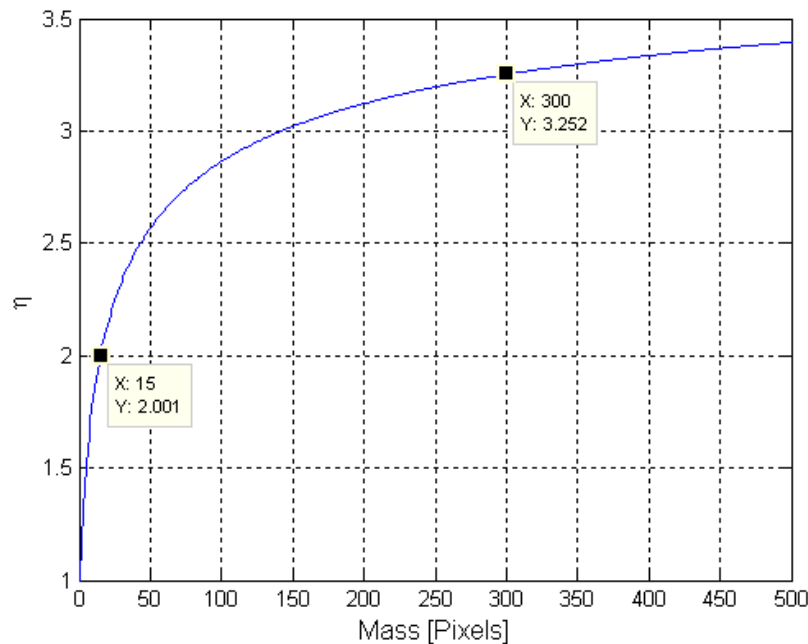


Figure 5.13: The ratio, η , between the region growing algorithm's and the new algorithm's execution speed.

On the other hand, the region growing algorithm is more robust than the faster algorithm as it can handle light spots of any shape. The faster algorithm can handle any convex light spots (circles, ellipses, triangles, rectangles etc), but it returns an inaccurate centroid for any concave light spot, as shown in figure 5.14(a). For a light spot of this shape, the faster algorithm would only use the shaded region shown in figure 5.14(b) to determine the centroid, resulting in an inaccurate answer.

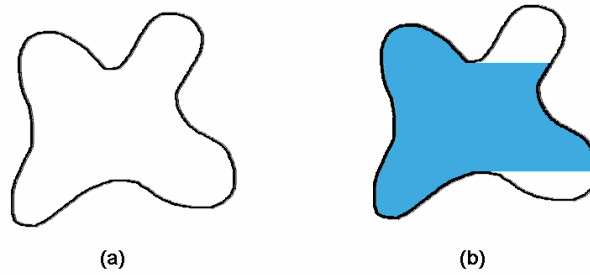


Figure 5.14: For the light spot shown in (a), the faster algorithm will only use the shaded region shown in (b) to determine the centroid.

5.6.2 Conclusion

For the original application of determining the centroids of stars, the region growing algorithm may be preferred due to its robustness. In this thesis the faster algorithm is used to ensure a higher sensor bandwidth, especially at close range. The smallest centroid mass in this project is about 22 pixels (at maximum range) and the largest mass is in the excess of 300 pixels at close proximity. Therefore the faster algorithm's bandwidth is more than 2 times faster than the region growing algorithm, as shown in figure 5.13. It is expected that all light spots will be convex (either circular or elliptical) so that the faster algorithm will work reliably.

Chapter 6

System Overview

An overview of the system is given in this chapter. From this the necessary hardware for the emulation can be determined.

6.1 Chaser Overview

Figure 6.1 shows a diagrammatic illustration of the chaser. All the lighter blocks represents software components, while the darker blocks represents hardware components.

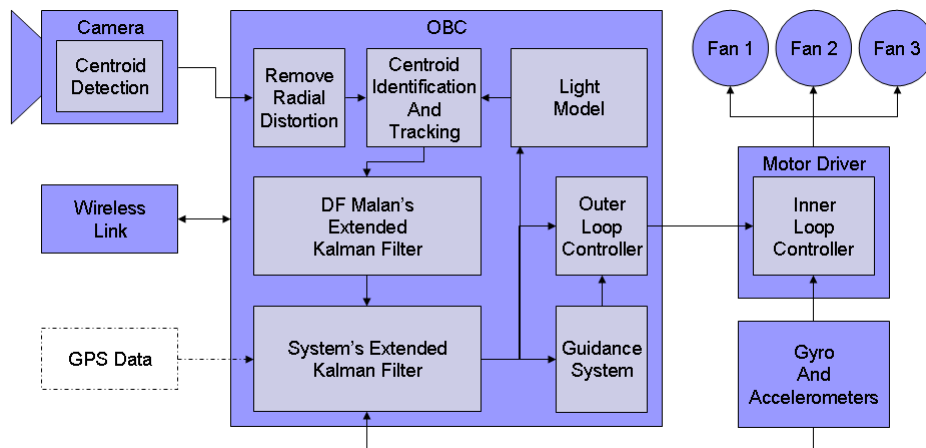


Figure 6.1: Overview of the chaser.

The chaser is emulated using a hovercraft. (From now on the term "chaser" and "hovercraft" will be used interchangeably.) Three fans (the minimum required) are used to produce accelerations along the chaser's x- and y-axis and an angular acceleration around the z-axis. It should be noted that the hovercraft's movement can be fully described using purely kinematic equations and all the dynamics in the system are caused by the actuator forces.

A motor driver board is required to control and drive the actuators. As illustrated in figure 6.1, the inner loop acceleration controllers are implemented on this board. The inner loop controller runs at 50 Hz and sends its accelerometer and gyro data to the Onboard Computer

(OBC) every 100 ms. In so doing, the inner loop controller provides the timebase for the outer loop controller.

The chaser is also equipped with a single camera unit. After an image is taken, the camera unit's microcontroller sends the detected centroids to the OBC for further processing.

The OBC receives the centroids from the camera unit and first removes the radial distortion, before passing the corrected centroids to the centroid identification and tracking unit. When the centroids are uniquely identified / matched, as described in chapter 5, the 2D-3D point pairs are sent to Malan's EKF. This EKF determines the target's relative position, orientation, velocity and angular rates. The system EKF takes the updated state vector of Malan's EKF (referred to as the camera measurement) and uses it to update its own states. Between camera measurements, accelerometer and gyro data is used by the system EKF for propagation. Based on the system EKF's states, the guidance system schedules the correct outer loop controller and also provides the necessary reference signals. Commands from the outer loop controller are then sent to the inner loop controller. The system EKF's state vector is also used by the light model to aid the centroid identification and tracking unit as described in chapter 5.

For reasons given in section 7.6, the GPS part of this thesis was discarded. Therefore it is shown in dashed lines.

Lastly, a wireless communications link is used to exchange data between the chaser and target.

6.2 Choice of Operating System for the Chaser's OBC

A total of 4 operating systems were considered for the chaser's OBC. Here is a short summary of them:

- **Windows / Linux**

Both these operating systems have layered architecture whereby the hardware is encapsulated by the operating system. Therefore the user process can only gain indirect access to the hardware using the interface provided by the operating system. (Windows API or Linux equivalent) This indirect access is not only tedious but also more complex to implement.

These operating systems also require much more system resources increasing the overall system overheads. Then, as these are multitasking operating systems, the user process shares the CPU with system processes.

Taking the above mentioned into account, it should be clear that real time execution is not guaranteed. (Supporters of these two operating systems normally assumes that the hardware is fast enough to ensure real time execution.)

Another point that should be considered is that these operating systems make use of paging. This is undesired as the Compact Flash used can only be written to a finite amount of time, so that paging drastically reduces the Compact Flash's lifespan. One must therefore ensure that this feature can be disabled.

- **DOS**

DOS is a very simple operating system and allows direct access to all hardware. This operating system has very little overheads as it has been designed to work efficiently on the original computers with very limited capabilities. Apart from this, the CPU is dedicated to the user process so that real time execution is easily achieved. Due to its simplistic nature, this operating system does not make use of paging.

- **AMX**

AMX is a real time operating system that supports multiple processes. It uses DOS for bootstrapping and then takes full control of the system. The user is provided with the operating system's source code (written in C) and then compiles it along with the user process' code. This results in a single executable that contains both the operating system and user application.

After considering the above, it was decided to use DOS due to its easy hardware access and efficiency. There was decided against AMX as its advanced capabilities are not required.

6.3 Target Overview

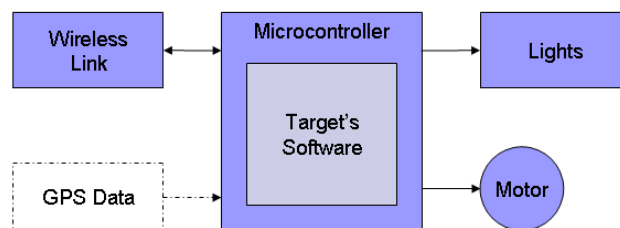


Figure 6.2: Overview of the target.

The target is a very simple system as seen from its diagram in figure 6.2. An imitation of a satellite is placed on a rotating platform. The lights and stepper motor speed (target's angular rate) is controlled by a microcontroller. Again the GPS unit is shown in dashed lines as it was discarded later.

6.4 Selection of the Sample Frequencies for the Inner and Outer Loop Controllers

It is advised by Franklin [7] that a minimum of 5 sample periods should fit into the rise time of a step response in order to ensure a relatively smooth response. To make provision for a minimum closed loop rise time of 100 ms, one requires a maximum sample period of 20 ms. Therefore the sample frequency for the inner loop, f_{s_i} , is selected as 50 Hz.

For anti-aliasing purposes, the inner loop sensors' bandwidth should not exceed the Nyquist frequency of 25 Hz. This implies that the anti-aliasing filters maximum cutoff frequency must

be less than 25 Hz. As shown in section 7.4, the actuators are modeled as first order low pass filters. If the closed loop system is also seen as a low pass filter and if provision is made that it can reach its final value in a minimum of 100 ms, then the maximum frequency in its output would be less than

$$\begin{aligned} f_c &= \frac{1}{2\pi(20 \times 10^{-3})} \\ &= 7.958 \text{ Hz} \end{aligned}$$

as it is generally assumed that a first order low pass filter reaches its final value in 5 time constants.

The anti-aliasing filter's cut off frequency was chosen at 15 Hz to minimise the additional phase lag to the plant model, while ensuring that the attenuation is about -8.87 dB at 25 Hz to minimise aliasing.

To choose the outer loop controller's sample frequency one should consider:

- The camera measurement will be available at an expected rate of about 2 Hz.
- The GPS measurements will be available at 4 Hz.
- To ensure that the inner loop and outer loop controllers does not influence each other, the outer loop controller should be at least 5 times slower than the inner loop controller.
- A faster sample frequency ensures more accurate propagation.

Taking the above into consideration, the outer loop sample frequency, f_{s_o} , was chosen as 10 Hz.

6.5 Conclusion

This chapter gave an overview of the system. From this it can be determined what hardware is required. The hardware is discussed in chapters 7 and 8. Refer to appendix D for photos of the chaser and the target.

Chapter 7

Hardware Implementation

This chapter discusses the most important hardware as well as the calibration and the modeling thereof. The camera will be discussed in detail in chapter 8. Refer to appendix D for photos of the hardware.

7.1 The Onboard Computer (OBC)

The CoreModule 420 of Ampro was selected as it provides the most compatible and cost effective solution for this application. Running at 133 MHz, this OBC consumes 7.5 Watts at full load and comes standard with 64 MB of RAM and 4 serial ports. It supports both Compact Flash and Disk-On-Chip. Compact flash was used in this project as it is much cheaper and more readily available than Disk-On-Chip. It is also compatible with MSDOS, the operating system to be used in this application.

Other OBCs that were considered were the PM-GX-466-R10, PM-1041 from Eagle, Diamond System's Prometheus, Tri-M's TMZ104 and Netcom's ASBC11. The TMZ104 emulates an x86 processor. Apart from the fact that one does not know how efficient this emulation process is, the TMZ104, like the PM-1041, only uses the expensive Disk-On-Chip. Netcom's ARM11 based ASBC11 is more aimed at the video processing market while the Prometheus processor is obsolete. The PM-GX-466-R10 requires Windows or Linux based drivers and will therefore not be fully compatible with a DOS environment.

7.2 Navigation and Attitude Sensors

The sensors are divided into two groups. Those that are used to update the extended Kalman filters and those used to propagate the extended Kalman filters.

7.2.1 Sensors used for updating

A brief comparison of the sensors that were considered is given here. Refer to Feshe [6] for an excellent comparison.

- **Radio Frequency (RF) sensors** measures the relative position of the target satellite by measuring the time of flight of a transmitted signal between the chaser and target satellite.

The relative velocity is measured by observing the Doppler shift. The main disadvantage of RF sensors is that they require large amounts of power as the transmission power is directly proportional to the fourth power of the distance between the chaser and target. (Given that the target does not have an active transponder.) These sensors are only suited for medium to long ranges as disturbances due to multipath effects and shadowing (where the RF signals are obstructed) prevents the use during close proximity manoeuvres.

- **Satellite Navigation Systems.** Two common satellite navigation systems are America's Global Positioning System (GPS) and Russia's Global Orbiting Navigation Satellite System (GLONASS). In the rest of this discussion there will only be referred to GPS, but the same holds for GLONASS.

A single GPS receiver can determine its absolute position in an Earth Centred, Earth Fixed (ECEF) reference frame. This is referred to as absolute satellite navigation.

For satellite docking and formation flying, one is more interested in the relative position and velocities of the satellites. With 2 GPS receivers, the more accurate relative satellite navigation is possible. Relative GPS (RGPS) requires that the one GPS receiver's data be sent to the other receiver via a communications link. The two GPS receivers' positions are then subtracted from one another resulting in a more accurate measurement of the relative distance between the satellites as common mode noise, such as atmospheric interference, common to both signals are canceled out. (This last statement is true given that the 2 receivers are within a few tens of kilometres from each other and that the receivers use the same navigation satellites.) The power consumption of the GPS receivers are very low, but as RGPS requires a communications link, the maximum distance is limited by the wireless communication link's range. (Note that this power requirement is still much lower than in the case of RF sensors, as the signal only has to be sent from one satellite to the other, instead of bouncing a signal off the other satellite.) Sadly, RGPS suffers from the same problems as RF sensors at close proximity. Therefore RGPS is also only suited for the medium to long range.

Feshe [6] states in his book, that at the time of writing, the accuracies of RGPS were 10 m for relative position and $0.05 \text{ m}\cdot\text{s}^{-1}$ for relative velocities.

Carrier-Differential Phase GPS (CDGPS) is an improvement on RGPS in terms of accuracy. It uses phase information to achieve accuracies of 0.02-0.05 m for relative position and $0.01 \text{ m}\cdot\text{s}^{-1}$ for relative velocities. Apart from that, it has the same requirements and limitations of RGPS. [8]

- The **Scanning Laser Range Finder** is an optical sensor and has the same working principle as RF sensors, but only uses infra-red light. (In other words, it just uses a different frequency in the electromagnetic spectrum.) Another difference is that the relative velocity is determined by differentiating the filtered relative position. Its main advantages is that it is suitable for the long to short range (1 meter to a few kilometers) and is less sensitive to the sun due to its very narrow beam width. Its biggest disadvantage is that it uses rotating mirrors. This results in a lower scanning rate (bandwidth) and a less reliable system as the bearings can be damaged by launch vibrations. Although it is less susceptible

to the sun, it is not immune against it. Lastly, this sensor also has an minimum range of about 1 m.

- **Camera.** As stated in chapter 1, the camera is considered to be the best overall sensor in terms of performance, mass and power consumption. It is mostly suited for the short range and has several advantages, such as:
 - The measurement accuracy improves as the relative distance decreases.
 - It can be used until capture.
 - There are no moving parts and therefore the camera is much more robust against launch vibrations.
 - It has a potentially high bandwidth. This is determined not only by the cameras frame rate, but also by the processing power and efficiency of the algorithms used.

Being an optical sensor with a large field of view, it has the disadvantage of being susceptible to several noise sources:

- The sun.
- Reflections of the targets surface. These reflections can be caused by either the sun or by the target satellite's own lights.
- Other stray light sources.

Of all the noise sources, the sun is the biggest threat. It practically radiates all light frequencies so that sharp filtering does not solve this problem.

GPS has mostly replaced RF sensors in the long and medium range, while camera sensors are predominately used in the short range. Therefore it was decided to use these two sensors in this thesis. By using an optical filter the influence of the stray light sources can be eliminated. To compensate for the sun, the docking manoeuvre must be executed in such a way to minimise the sun's influence. If this proves to be insufficient, the docking sequence must be restricted to the eclipse part of the orbit.

7.2.2 Sensors used for propagating

A dual axis accelerometer and a single axis rate gyro are used as the hovercraft only has 3 degrees of freedom. (Translation and rotation within a 2 dimensional plane.) Analog Devices' ADXL203 accelerometer and ADXRS401 rate gyro were chosen due to their low cost, availability and acceptable noise characteristics. Apart from this, these sensors also have a history of use in the ESL. [1]

7.3 The Accelerometers and Gyros

7.3.1 Calibration of the accelerometers and rate gyro

Calibration is done to ensure that the corrected measurement is used as an accurate representation of the measured quantity. This correction is required due to the use of non-ideal compo-

nents. For example: The accelerometer's gain differs from the value specified in the datasheet, the low pass filter's passband gain is not exactly unity and the reference voltage may not be exactly 5 V.

7.3.1.1 Accelerometers

The accelerometers are calibrated using the model of Bijker [1]:

$$\begin{bmatrix} \ddot{x}_{AD} \\ \ddot{y}_{AD} \end{bmatrix} = G_{acc} \left(\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} \ddot{x}_m \\ \ddot{y}_m \end{bmatrix} + \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \right) \quad (7.3.1)$$

where G_{acc} is the theoretical gain from the accelerometer to ADC output, \mathbf{C} the cross coupling matrix and \mathbf{D} the bias vector. The accelerations measured are \ddot{x}_m and \ddot{y}_m , and the output of the ADCs are \ddot{x}_{AD} and \ddot{y}_{AD} . As the two accelerometers are manufactured in the same package with an alignment error of less than 0.1° , it is assumed that there is no cross coupling between them. ($c_{12} = c_{21} = 0$) Also, define:

$$c_{11} = \frac{1}{K_x}$$

$$c_{22} = \frac{1}{K_y}$$

so that K_x and K_y are the calibration constants with which a measurement must be multiplied in order to correct it.

Now calibration is done using the method from Groenewald [9], where the gravitational acceleration is assumed to be $g = 9.81 \text{ m.s}^{-2}$. The following is done for each accelerometer axis:

1. The accelerometer is turned so that the measured acceleration is a maximum. (In other words +1g) This measurement is averaged over 1 minute and then recorded.
2. Then the accelerometer is turned so that the measured acceleration is a minimum. (In other words -1g) This measurement is also averaged over 1 minute and then recorded.

It is assumed that the bias remains constant over the time interval of the experiment for each axis. Next, the difference between the maximum and minimum values of each axis is calculated in order to remove the bias. For the x-axis, for example:

$$\begin{aligned} \Delta \ddot{x}_{AD} &= \ddot{x}_{AD}^+ - \ddot{x}_{AD}^- \\ &= G_{acc} \left(\frac{g}{K_x} + D_1 \right) - G_{acc} \left(\frac{-g}{K_x} + D_1 \right) \\ &= G_{acc} \left(\frac{2g}{K_x} \right) \end{aligned}$$

From this can be seen that:

$$K_x = G_{acc} \left(\frac{2g}{\Delta\ddot{x}_{AD}} \right) \quad (7.3.2)$$

$$K_y = G_{acc} \left(\frac{2g}{\Delta\ddot{y}_{AD}} \right) \quad (7.3.3)$$

The constant, G_{acc} , can be calculated from the data sheets to be:

$$G_{acc} = \frac{2^{12}}{5g}$$

Three tests were done and the results were average in order to obtain $\Delta\ddot{x}_{AD}$ and $\Delta\ddot{y}_{AD}$. The results are summarised in table 7.1.

Test	\ddot{x}_{AD}^+	\ddot{x}_{AD}^-	$\Delta\ddot{x}_{AD}$	Test	\ddot{y}_{AD}^+	\ddot{y}_{AD}^-	$\Delta\ddot{y}_{AD}$
1	2888.9	1259.3	1629.6	1	2873.6	1232.5	1641.1
2	2889.0	1256.1	1632.9	2	2872.9	1232.9	1640.0
3	2888.6	1259.2	1629.4	3	2872.4	1232.8	1639.6
Average			1630.63	Average			1640.23

Table 7.1: The results from the calibration experiments.

From the table, the average for $\Delta\ddot{x}_{AD}$ is 1630.63 and the average for $\Delta\ddot{y}_{AD}$ is 1640.23. Using equations 7.3.2 and 7.3.3 gives K_x and K_y to be 1.0048 and 0.9989, respectively.

7.3.1.2 Gyros

The angular rate gyro is mounted on the same board as the accelerometers. It is assumed that this gyro's axis is perpendicular to the accelerometers' axes and the gyro is then calibrated as such. Therefore, the gyro's calibration constant will include the effect of a non perpendicular axis (if it is the case) and automatically compensate for it.

There are 2 common methods to calibrate a rate gyro. The first is to use a rate table to apply a known angular rate and compare this known angular rate with the measured value.

The second method is to integrate the gyro's output while the gyro is turned through a known angle. Calibration is then done by comparing the measured (calculated) angle with the know angle.

As, at the time of calibration, access to a rate table was not possible, the second method was used using the same principle as used to calibrate the accelerometers.

Similar to the model used for the accelerometers, the model for the gyro calibration is:

$$\dot{\theta}_{AD} = G_{gyro} \left(\frac{\dot{\theta}_m}{K_\theta} + D_3 \right)$$

Where $\dot{\theta}_{AD}$ is the output of the ADC, $\dot{\theta}_m$ is the angular rate being measured, K_θ is the gyro's calibration constant and D_3 is the gyro's bias. Note that $\dot{\theta}_m$ is in degrees per second. Again from the datasheets:

$$G_{gyro} = \left(\frac{2^{12}}{5} \right) \left(\frac{15}{1000} \right)$$

A jig was made on which the gyro and accelerometer board was mounted. This jig allows the gyro to be turned through a known angle of 90° . The calibration process¹ is done as follows:

1. First measure the gyro's output while the gyro is stationary. This measurement is averaged over 1 minute to give the bias as $\dot{\theta}_m = 0$. It is assumed that the bias remains constant over the time interval of this experiment.
2. Now, measure and record the gyro's output while the gyro is repeatedly turned from 0° to 90° and back over a 30 second interval.

Then the result, of subtracting the measured bias from the measured angular rate, is integrated:

$$\begin{aligned} \theta_{AD} &= \int (\dot{\theta}_{AD} - G_{gyro}D_3) dt \\ &= \int \left(G_{gyro} \left(\frac{\dot{\theta}_m}{K_\theta} + D_3 \right) - G_{gyro}D_3 \right) dt \\ &= \left(\frac{G_{gyro}}{K_\theta} \right) \int (\dot{\theta}_m) dt \end{aligned}$$

Dividing this equation by G_{gyro} gives the angle in degrees:

$$\begin{aligned} \theta &= \left(\frac{1}{K_\theta} \right) \int (\dot{\theta}_m) dt \\ &= 90^\circ \end{aligned} \tag{7.3.4}$$

Figure 7.1 shows a typical result of this integration process. Next, the average range of θ is determined by:

$$\overline{\Delta\theta} = \frac{1}{N} \sum_{n=1}^N (\theta_{max_n} - \theta_{min_n})$$

Where N is the number of maxima-minima pairs. (Figure 7.1 has three pairs shown as stars.) The average range of θ is given by $\overline{\Delta\theta}$, and the maximum and minimum values are given by θ_{max_n} and θ_{min_n} , respectively. Using the average range, $\overline{\Delta\theta}$, and equation 7.3.4, the value of K_θ can be calculated by:

$$K_\theta = \frac{90}{\overline{\Delta\theta}} \tag{7.3.5}$$

This test was done 3 times and the average of these test results was used to calculate K_θ . For brevity, only the results of the 3rd test, that corresponds with figure 7.1, is displayed in table 7.2.

¹A sampling frequency of 50 Hz was used.

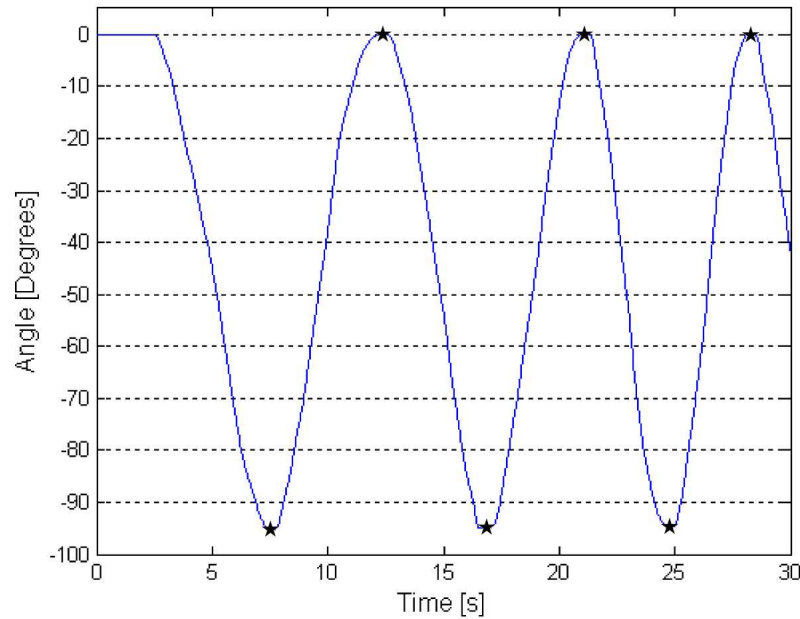


Figure 7.1: The result of the integration process.

Pair (n)	θ_{max_n} [°]	θ_{min_n} [°]	$\theta_{max_n} - \theta_{min_n}$ [°]
1	0.0996	-95.2500	95.3496
2	0.3766	-94.9800	95.2666
3	0.3086	-94.8000	95.1086
Average			95.24

Table 7.2: Results of the 3rd gyro calibration test.

The average range of θ over the 3 test is 95.10° . Therefore from equation 7.3.5:

$$\begin{aligned}
 K_\theta &= \frac{90}{\overline{\Delta\theta}} \\
 &= 0.9464
 \end{aligned}$$

7.3.2 Modeling of the accelerometers' and rate gyro's noise

Any practical sensor's output is the superposition of the actual value and a random noise component. The purpose of this section is to model the random component of the sensor's output.

In his thesis, Hou [12] relates the Allan variance method of data analysis (originally used for the study of oscillator stability) to 7 noise sources, where the random component is the superposition of these noise sources. The advantage of this approach is that it allows finer, simpler characterisation of the error sources and their contribution to the overall noise. Of these 7 noise sources, this thesis only uses 2, as their influence is dominant. [1] These noise sources are:

- Angle (velocity) random walk. Angle (velocity) random walk is the measurement noise

on the output of the sensor that causes a random walk in the angle (velocity) after the output of the sensor is integrated over time.

- Rate random walk. Rate random walk is the instability of the output of the sensor. The integral of this noise source causes the bias of the sensor to drift.

To visualise this refer to figures 7.5 and 7.6. The output of the sensors was recorded for about 30 minutes and then processed using the Allan variance method. This method has been thoroughly described by Bijker [1], Herselman [11] and Hou [12].

7.3.2.1 Accelerometers

Figures 7.2 and 7.3 shows the Allan deviation plotted against the interval time for each accelerometer. The value of the velocity random walk spectral density coefficient, Q , and the

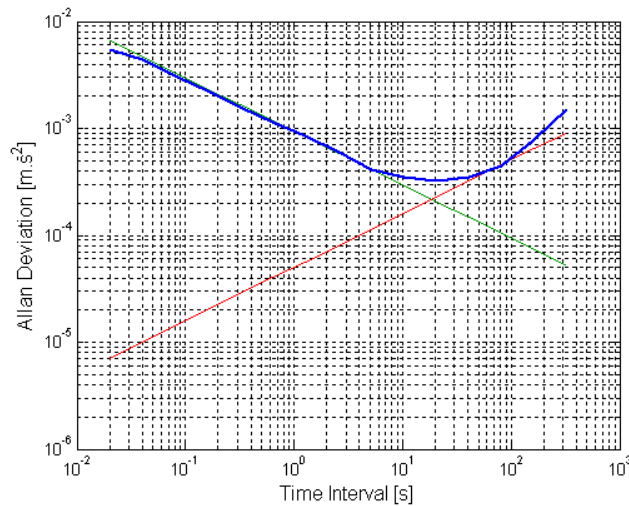


Figure 7.2: The Allan deviation for the x-axis accelerometer.

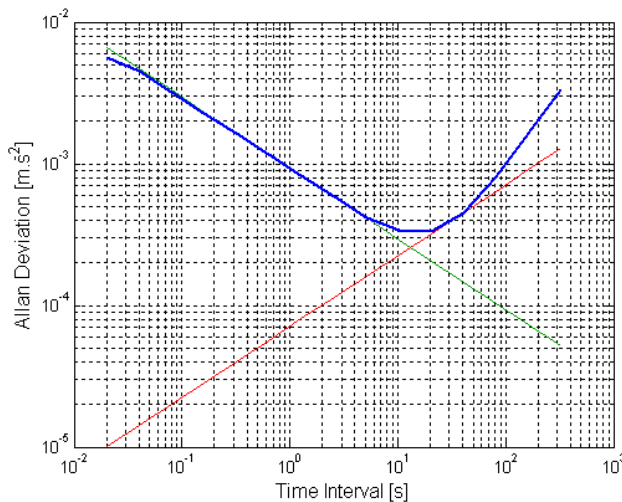


Figure 7.3: The Allan deviation for the y-axis accelerometer.

rate random walk spectral density coefficient, K , can be obtained directly from the graph. The value of Q can be obtained where the line with the negative slope crosses $T = 1$ and the value of K can be obtained where the line with the positive slope crosses $T = 3$. These values are presented in table 7.3. From these values, the noise's power spectral densities or, equivalently, the covariances can be determined. Equation 7.3.6 gives the relationship between Q and the covariance of the velocity random walk noise component. [1]

$$\sigma_{VRW}^2 = 2 \int_{f_1}^{f_2} Q^2(f) df \quad (7.3.6)$$

The covariance for the rate random walk, σ_{RRW}^2 , is also given by equation 7.3.6, except that Q is substituted by K . Table 7.4 shows the resulting standard deviations.

Accelerometer	Q [$\text{m.s}^{-2}.\text{Hz}^{-1/2}$]	K [$\text{m.s}^{-3}.\text{Hz}^{-1/2}$]
X-axis	9.3325×10^{-4}	8.6808×10^{-5}
Y-axis	9.3325×10^{-4}	1.2262×10^{-4}

Table 7.3: Noise parameters of the accelerometers.

Accelerometer	σ_{VRW} [m.s^{-2}]	σ_{RRW} [m.s^{-3}]
X-axis	5.11×10^{-3}	125.311×10^{-6}
Y-axis	5.11×10^{-3}	173.411×10^{-6}

Table 7.4: The standard deviations of the accelerometers' noise components.

From table 7.4 it can be seen that the accelerometers have very good noise characteristics. But these are the noise covariances when the accelerometers are stationary. Unfortunately, the hovercraft fans, especially the lift fan, creates fairly intense mechanical vibrations and the accelerometers are very susceptible to these mechanical vibrations. Even after the accelerometers have been soft mounted and their outputs filtered by analog and digital filters, the noise in the accelerometers outputs has a standard deviation of about 0.06 m.s^{-2} .

7.3.2.2 Gyros

Exactly the same procedure is followed for the rate gyro. The only difference is that Q is now called the angle random walk spectral density coefficient and the corresponding covariance is σ_{ARW}^2 . (The relationship between Q and σ_{ARW}^2 , and K and σ_{RRW}^2 is also given by equation 7.3.6.) Figure 7.4 shows the Allan deviation plotted against the interval time for the gyro and tables 7.5 and 7.6 summarises the results.

The gyro has very good noise characteristics, as can be seen from table 7.6. Fortunately these noise characteristics remain mostly unaffected by the mechanical vibrations.

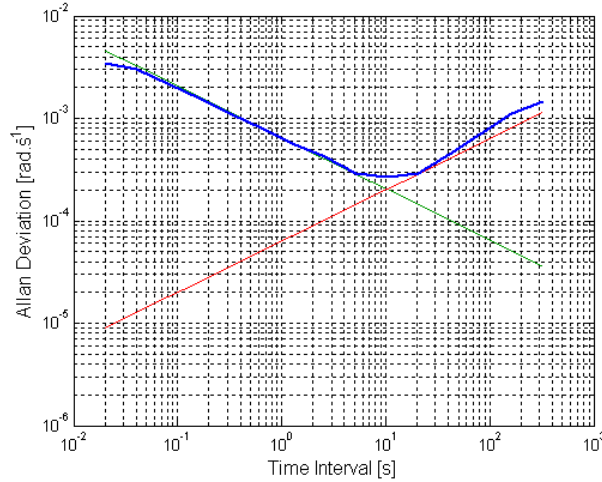


Figure 7.4: The Allan deviation for the z-axis rate gyro.

Gyro	Q [rad.s ⁻² .Hz ^{-1/2}]	K [rad.s ⁻³ .Hz ^{-1/2}]
Z-axis	6.4565×10^{-4}	1.0929×10^{-4}

Table 7.5: Noise parameters of the gyros.

Gyro	σ_{ARW} [rad.s ⁻²]	σ_{RRW} [rad.s ⁻³]
Z-axis	3.536×10^{-3}	154.559×10^{-6}

Table 7.6: The standard deviations of the gyro's noise components.

7.3.2.3 Conclusion

The noise variances obtained in this section can now be used to design the inner loop estimators in chapter 10.

7.3.3 Full accelerometer and gyro models

By combining the results from sections 7.3.1 and 7.3.2, the full models of the accelerometers and gyro is obtained. Figures 7.5 and 7.6 shows an implementation of these models.

7.3.4 Comparing the Allan variance results with the values from the data sheets

The variances σ_{ARW} and σ_{VRW} , obtained using the Allan variance method, can also be calculated from the data sheets. This section gives a quick comparison of the values obtained using this approach.

From the gyro's data sheets, it is estimated that its noise density is 0.94868×10^{-6} V.Hz^{-1/2} rms, so the the noise standard deviation is:

$$\begin{aligned}\sigma_{ARW} &= \sqrt{(0.94868 \times 10^{-6})^2(15)} \\ &= 3.674 \times 10^{-3} \text{ V}\end{aligned}$$

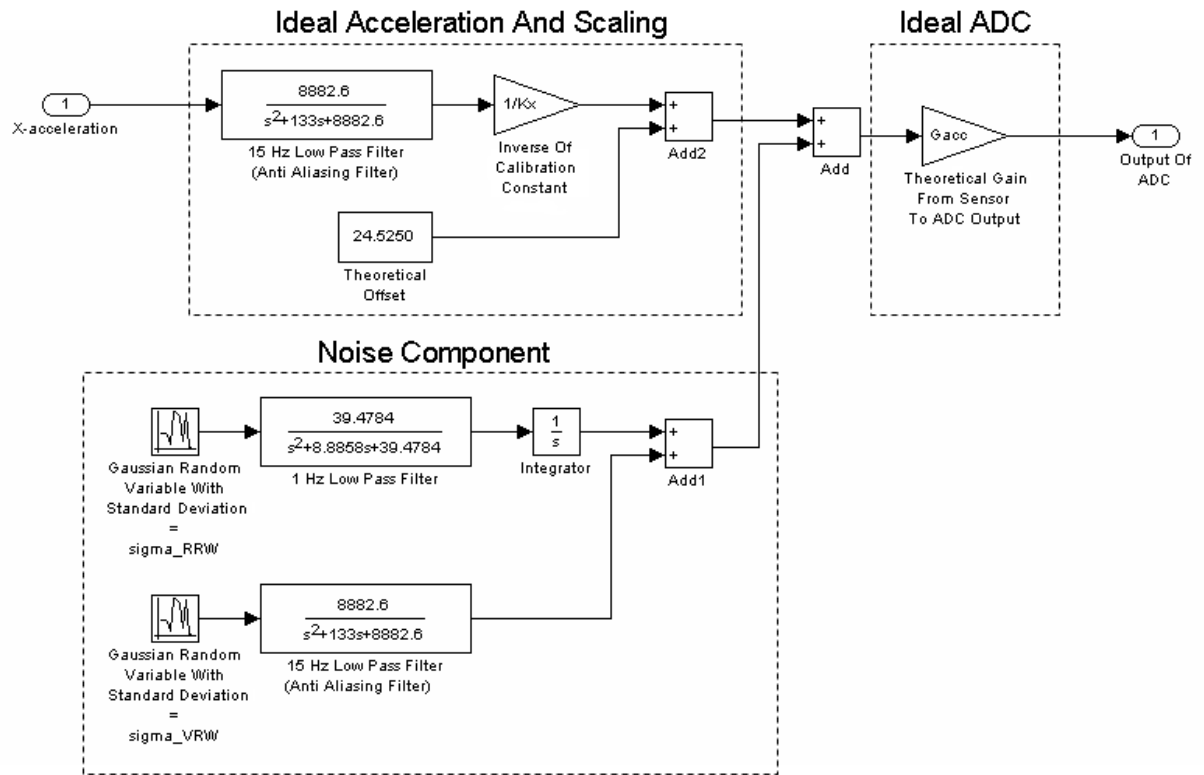


Figure 7.5: Full accelerometer model. (X-axis accelerometer shown.)

This standard deviation’s unit must be converted to $\text{rad}\cdot\text{s}^{-1}$ in order to compare it:

$$\begin{aligned} \sigma_{ARW} &= (3.674 \times 10^{-3}) \left(\frac{1000}{15} \right) \left(\frac{\pi}{180} \right) \\ &= 4.275 \times 10^{-3} \text{ rad}\cdot\text{s}^{-1} \end{aligned}$$

The accelerometer’s noise density is given as $110 \mu\text{g}\cdot\text{Hz}^{-1/2}$, so that:

$$\begin{aligned} \sigma_{VRW} &= \sqrt{(110 \times 10^{-6})^2(15)} \\ &= 426.0 \times 10^{-6} \text{ V} \end{aligned}$$

Again, in order to compare this standard deviation to the Allan variance method’s result, its unit has to be changed to $\text{m}\cdot\text{s}^{-2}$:

$$\begin{aligned} \sigma_{VRW} &= (426.0 \times 10^{-6}) (9.81) \\ &= 4.179 \times 10^{-3} \text{ m}\cdot\text{s}^{-2} \end{aligned}$$

Table 7.7 lists the results obtained using the Allan variance method and the results from the data sheets. It can be seen that the two methods gives similar results.

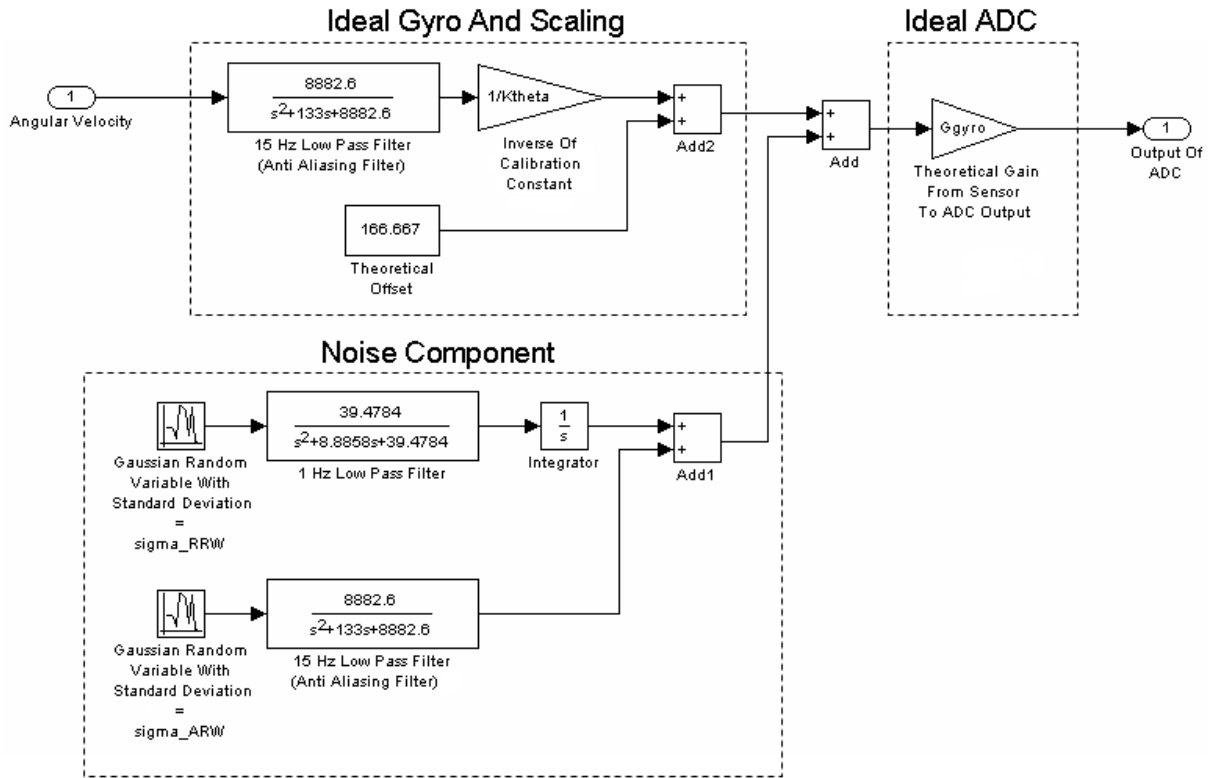


Figure 7.6: Full gyro model. (Note: The input angular rate is in rad.s^{-1})

Method	Gyro σ_{ARW} [rad.s^{-1}]	Accelerometer σ_{VRW} [m.s^{-2}]
Allan Variance	3.536×10^{-3}	5.11×10^{-3}
Data sheets	4.275×10^{-3}	4.179×10^{-3}

Table 7.7: Comparing the results from the two different methods.

7.3.5 Aligning the accelerometer and gyro board's axis with the chaser's axis

After the accelerometer and gyro board was mounted on the hovercraft, the orientation of these sensors' axis in the chaser's body axis had to be determined. The sensor board's axis system is show in figure 7.7. Considering figure 7.7, one expects the sensor board's orientation in the

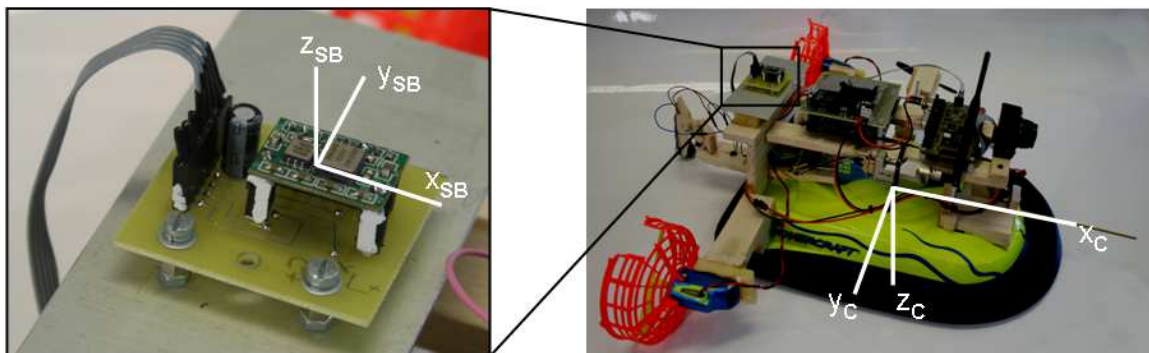


Figure 7.7: The sensor board.

chaser's body axis to be described by a rotation of 180° around the x-axis. (Note that for the axis system to be a right handed system as shown, the gyro output's sign has to be inverted. This is done in software.) This orientation was determined by using the accelerometers.

Again, consider equation 7.3.1:

$$\begin{bmatrix} \ddot{x}_{AD} \\ \ddot{y}_{AD} \end{bmatrix} = G_{acc} \left(\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} \ddot{x}_m \\ \ddot{y}_m \end{bmatrix} + \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \right) \quad (7.3.7)$$

But now the reference acceleration will be specified in the chaser's body axis and has to be transformed to the sensor board's axis using a DCM, $\mathbf{T}_{CHR \rightarrow SB}$. Due to the absence of a z-axis accelerometer, the last row of the DCM matrix can not be used. So if

$$\mathbf{T}_{CHR \rightarrow SB} = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{t}_3 \end{bmatrix}$$

let the partial DCM, \mathbf{T}_p be defined as

$$\mathbf{T}_p = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{bmatrix}$$

so that

$$\begin{bmatrix} \ddot{x}_m \\ \ddot{y}_m \end{bmatrix} = \mathbf{T}_p \ddot{\mathbf{x}}_{ref}^{CHR} \quad (7.3.8)$$

Where $\ddot{\mathbf{x}}_{ref}^{CHR}$ is the reference acceleration in the chaser's body axis. To simplify notation, $\ddot{\mathbf{x}}_{ref}^{CHR}$ will be referred to as $\ddot{\mathbf{x}}_{ref}$ in the rest of this section. The gravity vector was used as reference. Each of the chaser's body axis (+X,-X,+Y,-Y,+Z,-Z) was aligned with the gravity vector. This was done using a weight on a string. When the string hangs still, it gives the direction of the gravity vector. The hovercraft was then rotated to align each of the six axis with the gravity vector. For each axis, both accelerometers' measurements were averaged over 1 minute and recorded.

Each positive and negative axis pair were done consecutively so that it can be assumed that the biases remain constant over the time interval of the experiment of a positive-negative axis pair.

Here follows the measurements obtained. (Note that the superscript '+' and '-' indicates that the gravity vector was aligned with a positive and negative chaser body axis, respectively.)

For the X-axis:

$$\begin{aligned}\ddot{\mathbf{x}}_{ref_1}^+ &= \begin{bmatrix} 9.81, & 0, & 0 \end{bmatrix}^T \\ \ddot{\mathbf{x}}_{ref_1}^- &= \begin{bmatrix} -9.81, & 0, & 0 \end{bmatrix}^T \\ \ddot{\mathbf{x}}_{AD_1}^+ &= \begin{bmatrix} 2887.19, & 2047.76 \end{bmatrix}^T \\ \ddot{\mathbf{x}}_{AD_1}^- &= \begin{bmatrix} 1255.33, & 2059.38 \end{bmatrix}^T\end{aligned}$$

For the Y-axis:

$$\begin{aligned}\ddot{\mathbf{x}}_{ref_2}^+ &= \begin{bmatrix} 0, & 9.81, & 0 \end{bmatrix}^T \\ \ddot{\mathbf{x}}_{ref_2}^- &= \begin{bmatrix} 0, & -9.81, & 0 \end{bmatrix}^T \\ \ddot{\mathbf{x}}_{AD_2}^+ &= \begin{bmatrix} 2047.92, & 1232.77 \end{bmatrix}^T \\ \ddot{\mathbf{x}}_{AD_2}^- &= \begin{bmatrix} 2062.29, & 2873.13 \end{bmatrix}^T\end{aligned}$$

For the Z-axis:

$$\begin{aligned}\ddot{\mathbf{x}}_{ref_3}^+ &= \begin{bmatrix} 0, & 0, & 9.81 \end{bmatrix}^T \\ \ddot{\mathbf{x}}_{ref_3}^- &= \begin{bmatrix} 0, & 0, & -9.81 \end{bmatrix}^T \\ \ddot{\mathbf{x}}_{AD_3}^+ &= \begin{bmatrix} 2080.22, & 2077.24 \end{bmatrix}^T \\ \ddot{\mathbf{x}}_{AD_3}^- &= \begin{bmatrix} 2060.6, & 2034.89 \end{bmatrix}^T\end{aligned}$$

Now the positive and negative axis' measurements are subtracted from each other to remove the biases. Define:

$$\begin{aligned}\Delta\ddot{\mathbf{x}}_{ref_i} &= \ddot{\mathbf{x}}_{ref_i}^+ - \ddot{\mathbf{x}}_{ref_i}^- \\ \Delta\ddot{\mathbf{x}}_{AD_i} &= \ddot{\mathbf{x}}_{AD_i}^+ - \ddot{\mathbf{x}}_{AD_i}^-\end{aligned}$$

Where $i = 1, 2$ and 3 . Then all the measurements are packed in matrices:

$$\Delta\ddot{\mathbf{X}}_{ref} = \begin{bmatrix} \Delta\ddot{\mathbf{x}}_{ref_1} & \Delta\ddot{\mathbf{x}}_{ref_2} & \Delta\ddot{\mathbf{x}}_{ref_3} \end{bmatrix} \quad (7.3.9)$$

$$\Delta\ddot{\mathbf{X}}_{AD} = \begin{bmatrix} \Delta\ddot{\mathbf{x}}_{AD_1} & \Delta\ddot{\mathbf{x}}_{AD_2} & \Delta\ddot{\mathbf{x}}_{AD_3} \end{bmatrix} \quad (7.3.10)$$

Combining equations 7.3.7, 7.3.8, 7.3.9 and 7.3.10:

$$\Delta\ddot{\mathbf{X}}_{AD} = G_{acc} \mathbf{CT}_p \Delta\ddot{\mathbf{X}}_{ref}$$

Everything is known except for \mathbf{T}_p so that \mathbf{T}_p can be solved in a least squares sense:

$$\mathbf{T}_p = \begin{bmatrix} 1.0007 & -0.0088 & 0.0120 \\ -0.0071 & -1.0001 & 0.0258 \end{bmatrix}$$

From this the Euler angles (using the Euler 3-2-1 convention) can be determined:

$$\begin{aligned} \psi &= \arctan\left(\frac{T_{12}}{T_{11}}\right) \\ &= -0.50453^\circ \end{aligned}$$

$$\begin{aligned} \theta &= \arcsin(-T_{13}) \\ &= -0.68936^\circ \end{aligned}$$

$$\begin{aligned} \phi &= \arcsin\left(\frac{T_{23}}{\cos(\theta)}\right) \\ &= -178.0825^\circ \end{aligned}$$

Where T_{ij} is the element in the i th row and j th column of matrix \mathbf{T}_p . The above mentioned results means that the sensor boards axis orientation in the chasers body axis is described by a rotation around the x-axis of -178.0825° , follow by a rotation around the y-axis of -0.68936° and lastly a rotation of -0.50453° around the z-axis. This compares well to what was expected at the beginning of this section.

These angles can be used to reconstruct the whole DCM. After the matrix was made orthonormal, the result is:

$$\mathbf{T}_{CHR \rightarrow SB} = \begin{bmatrix} 0.9991 & -0.0088 & 0.0120 \\ -0.0071 & -0.9987 & 0.0258 \\ 0.0123 & 0.0333 & -0.9998 \end{bmatrix}$$

7.4 Actuators

Three fans are used as actuators in order to produce simultaneous translation along both the hovercraft's x- and y-axis as well as a rotation around the z-axis. These fans are controlled by the motor driver board. (Refer to section 7.5)

There is also a fourth fan that is used to pressurise the hovercraft's air cushion. The original DC motor could not take the strain caused by the additional weight of all the hardware. It was therefore replaced by a brushless DC motor, a Park 180 Outrunner, used in remote controlled models. This motor requires a special Electronic Speed Control (ESC) unit that receives its reference from a RF receiver. A small, 8-pin PIC12F675 is used to mimic this receiver. In an attempt to mitigate the mechanical vibrations cause by the imperfect prop, an additional ball bearing was inserted to support the lift fan's prop. The rest of the discussion on the actuators does not refer to this lift fan.

7.4.1 Modeling of the actuators

Each actuator is described by the fairly simplistic model shown in figure 7.8. The non-linear transfer function block is the curve that relates the PWM modulation module's reference value, n_{duty} , to the output force, F , produced by the fan. Note that the PWM module is configured in such a way that a duty cycle of 0-100% is represented by a value of 0-1000. A first order low pass filter is used to model the fan's dynamics. Due to the systems low bandwidth, only the slow mechanical time constant is taken into consideration. (The much faster electrical time constant is ignored.) Although the non-linear transfer function should include saturation, it is enforced by the saturation block.

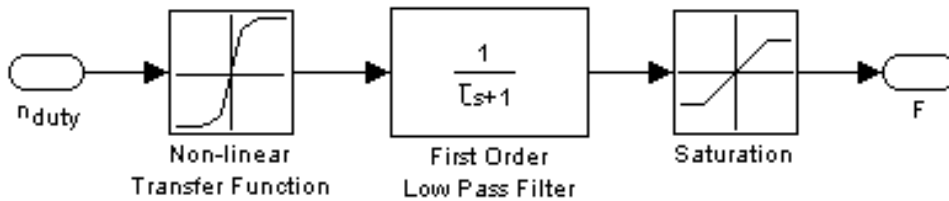


Figure 7.8: Motor model.

7.4.2 Calibration of the actuators

In this section, calibration implies the determination of the non-linear transfer curve that relates the duty cycle value, n_{duty} , to the fan's output force, F , as well as the motor's mechanical time constant, τ . During the calibration process the same motor driver board, that is used in the hovercraft, is used. Therefore, the transfer curve also includes all the effects of the motor driver.

The two fans whose output force vectors are parallel to the hovercraft's x-axis, are identical (same motor and prop) and will be collectively referred to as "Fan Type 1". The third fan whose output force vector runs parallel to the hovercraft's y-axis is referred to as "Fan Type 2".

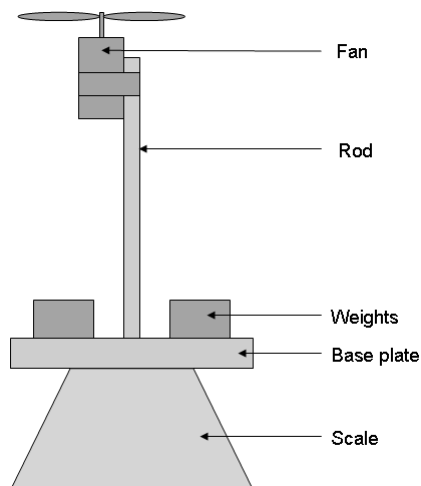


Figure 7.9: Actuator calibration setup.

Figure 7.9 shows the setup used to calibrate the fans. The following procedure is then followed to calibrate each fan type. Each fan is attached to one side of a rod. The other side of the rod is fitted in a base plate. This whole setup is then placed on a sensitive scale and weights are placed on the base plate to steady the setup. A very long rod of 1 m was used to ensure that airflow created by the fan does not influence the scale measurements.

The initial scale measurement is then recorded with the fan switched off ($n_{duty} = 0$). Thereafter, the duty cycle is systematically increased, in both the forward and reverse direction, while at each step the new scale reading is recorded.

For convenience, define that a positive duty cycle value causes a downward force vector (increasing the scale measurement) and that a negative duty cycle value causes an upward force vector (decreasing the scale measurement). Also let the downward force vector be represented by a positive force value as shown in figure 7.10.

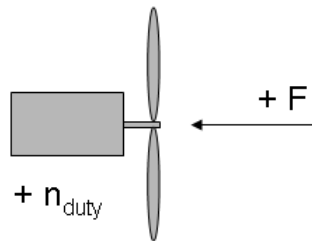


Figure 7.10: Direction of positive force value.

After this process, the changes in the scale measurements are calculated and converted to Newtons. Then a Least Squares approach is used to fit a piecewise defined function through the empirical data. The resulting transfer graphs (solid line) for each fan type along with the empirical values (circles) are shown in figures 7.11 and 7.12.

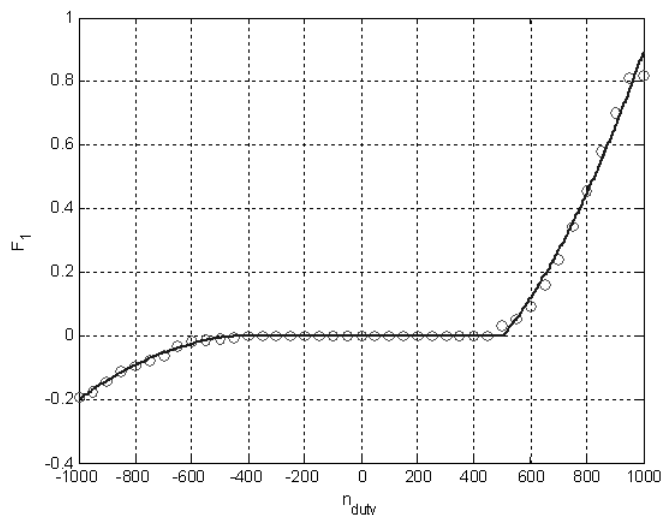


Figure 7.11: Fan type 1 transfer curve.

Equation 7.4.1 describes the transfer curve of the type 1 fan and equation 7.4.2 gives the transfer curve of the type 2 fan.

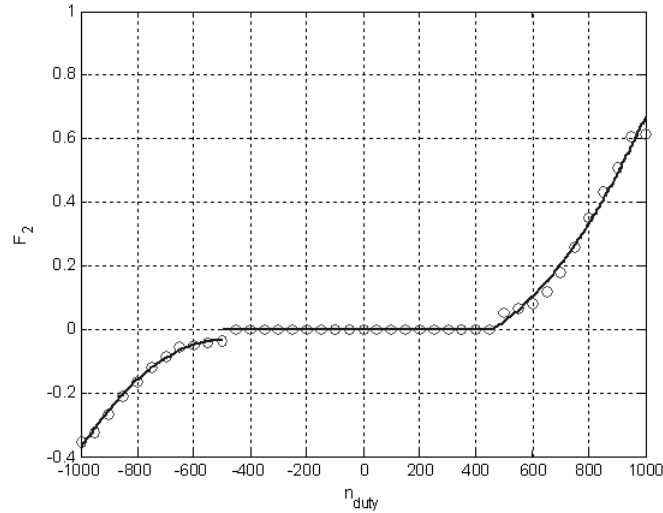


Figure 7.12: Fan type 2 transfer curve.

$$F_1(n_{duty_1}) = \begin{cases} (-5.051 \times 10^{-7})(n_{duty_1})^2 - 0.0003711(n_{duty_1}) - 0.06705 & , -1000 < n_{duty_1} < -414 \\ 0 & , -414 < n_{duty_1} < 504 \\ (1.425 \times 10^{-6})(n_{duty_1})^2 - 0.0003496(n_{duty_1}) - 0.1858 & , 504 < n_{duty_1} < 1000 \end{cases} \quad (7.4.1)$$

$$F_2(n_{duty_2}) = \begin{cases} (-1.28 \times 10^{-6})(n_{duty_2})^2 - 0.001236(n_{duty_2}) - 0.33 & , -1000 < n_{duty_2} < -500 \\ 0 & , -500 < n_{duty_2} < 460 \\ (1.286 \times 10^{-6})(n_{duty_2})^2 - 0.0006468(n_{duty_2}) - 0.0262 & , 460 < n_{duty_2} < 1000 \end{cases} \quad (7.4.2)$$

All that remains now is to estimate each motor's mechanical time constant. This is done by applying a 6 V step to the fan while measuring the motor's current by means of a low resistance series resistor. The current decays exponentially from a maximum value to a lower final value as the fan reaches its final speed and output force. From this graph the mechanical time constant can be estimated.

The measured fan currents for each fan type is shown by the light gray line in figure 7.13 (a) and (b) for fan type 1 and fan type 2, respectively.

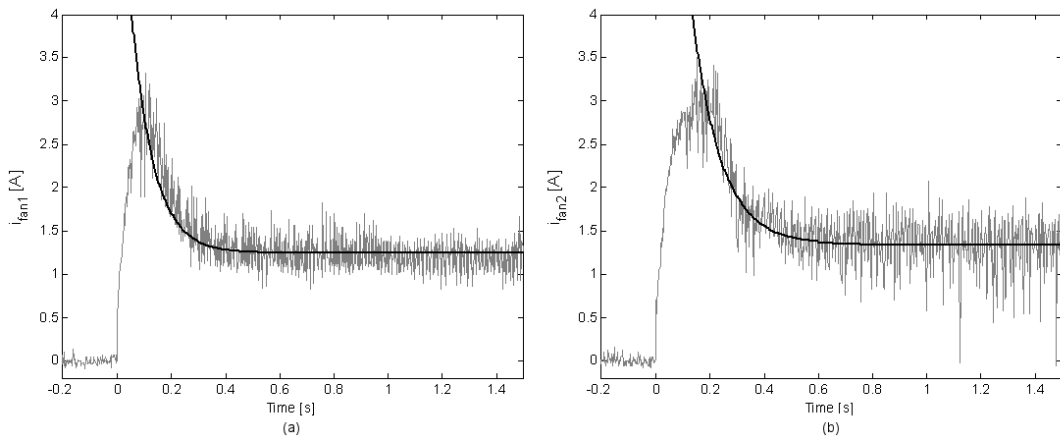


Figure 7.13: The fan currents. (a) Fan 1's current. (b) Fan 2's current.

To determine the time constant, the measured current was first filtered and then the following equation was fitted through the filtered data:

$$i = ce^{-t/\tau} + k \quad (7.4.3)$$

where i is the measured current, k is the fan current's final value, τ is the mechanical time constant and c is a scaling factor.

Equation 7.4.3 can be rewritten as:

$$\ln(i - k) = \ln(c) - \frac{t}{\tau}$$

Letting $\hat{c} = \ln(c)$ and $\hat{\tau} = 1/\tau$, then

$$[\ln(i - k)] = [1 \quad -t] \begin{bmatrix} \hat{c} \\ \hat{\tau} \end{bmatrix} \quad (7.4.4)$$

Now, two points from the filtered data are substituted into equation 7.4.4. If the two points are given by (t_1, i_1) and (t_2, i_2) , then:

$$\begin{bmatrix} \ln(i_1 - k) \\ \ln(i_2 - k) \end{bmatrix} = \begin{bmatrix} 1 & -t_1 \\ 1 & -t_2 \end{bmatrix} \begin{bmatrix} \hat{c} \\ \hat{\tau} \end{bmatrix} \quad (7.4.5)$$

Using least squares \hat{c} and $\hat{\tau}$ can be estimated, and from this c and τ can be determined.

From fan 1's graph, the final value, k , can be estimated as 1.24 A. Using the two points (0.114, 2.593) and (0.328, 1.338) in equation 7.4.5 results in:

$$\begin{bmatrix} 0.3023 \\ -2.3228 \end{bmatrix} = \begin{bmatrix} 1 & -0.1140 \\ 1 & -0.3280 \end{bmatrix} \begin{bmatrix} \hat{c} \\ \hat{\tau} \end{bmatrix} \quad (7.4.6)$$

Solving equation 7.4.6, gives $\hat{c} = 1.7007$ and $\hat{\tau} = 12.2669$ and from this that $c_{fan1} = 5.478$ A and $\tau_{fan1} = 0.0815$ s. This estimated curve is plotted as the thick black line in figure 7.13 (a).

Similarly, from fan 2's graph, the final value, k , can be estimated as 1.34 A. Using the two points (0.2, 2.808) and (0.478, 1.444) in equation 7.4.5 results in:

$$\begin{bmatrix} 0.3839 \\ -2.2634 \end{bmatrix} = \begin{bmatrix} 1 & -0.2000 \\ 1 & -0.4780 \end{bmatrix} \begin{bmatrix} \hat{c} \\ \hat{\tau} \end{bmatrix} \quad (7.4.7)$$

Solving equation 7.4.7, gives $\hat{c} = 2.2884$ and $\hat{\tau} = 9.5225$ and from this that $c_{fan2} = 9.8592$ A and $\tau_{fan2} = 0.1050$ s. This estimated curve is plotted as the thick black line in figure 7.13 (b).

By using the model in figure 7.8 and the results from this subsection, the fans can be modeled and simulated. This in turn is used to design the inner loop controller as discussed in chapter 10.

7.4.3 Linearising the actuators

From figures 7.11 and 7.12 can be seen that the actuators are non-linear. This nonlinearity can be compensated for by using the inverse of these transfer curves. To determine these inverse curves, n_{duty} is plotted against F and a Least Squares approach is used again to fit a piecewise defined function through this graph. The inverse functions for fan type 1 and fan type 2 is given by equation 7.4.8 and equation 7.4.9, respectively.

$$n_{duty1}(F_1) = \begin{cases} -276(F_1)^2 + 771.4(F_1) + 512.3 & , F_1 > 0 \\ 0 & , F_1 = 0 \\ -10300(F_1)^2 - 4664(F_1) + 459.6 & , F_1 < 0 \end{cases} \quad (7.4.8)$$

$$n_{duty2}(F_2) = \begin{cases} -560.2(F_2)^2 + 1118(F_2) + 491.8 & , F_2 > 0 \\ 0 & , F_2 = 0 \\ -3271(F_2)^2 - 2579(F_2) + 470.2 & , F_2 < 0 \end{cases} \quad (7.4.9)$$

Now, the control force commanded by the inner loop controller can be converted to the correct duty cycle value using the inverse curve and in doing so the non-linear fan is linearised, as illustrated in figure 7.14, given that the control force does not exceed the actuators limits.

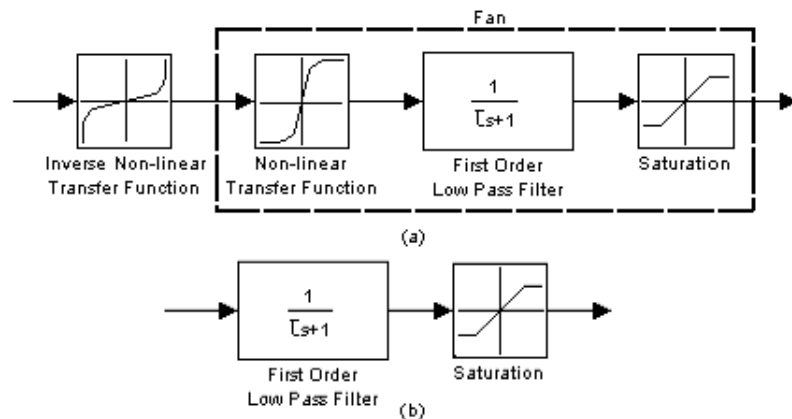


Figure 7.14: Linearising the actuator. (a) The non-linear fan with the inverse transfer curve. (b) The linearised fan model.

7.5 Motor Driver Board

Apart from the motor driver circuitry, the motor driver board also houses the inner loop controller and sends angular rate and acceleration data from the sensors back to the OBC.

The boards main features include:

- Two LT298 double H-bridge modules allowing a total of 4 fans to be controlled.
- A dsPIC30F5011 microcontroller that scans the sensor outputs, implements the inner loop controller and outputs the PWM control signals to the H-bridge modules.

- A RS232 communications interface.
- A sensor board equipped with a single axis gyro and dual axis accelerometer.

To prevent aliasing, second order butterworth low pass filters with a cut-off frequency of 15 Hz are used. All the filters have unity gain as no additional sensitivity is achieved by a higher gain. (From section 7.3.2, the accelerometers' and gyro's noise standard deviations are 520.897×10^{-6} V and 3.038×10^{-3} V, respectively, which amounts to 0.427 and 2.489 ADC units². Due to vibrations the accelerometer's standard deviation increases to 0.06 m.s⁻² or 5.01 ADC units.) The sensor board is mounted on an aluminium plate to increase the overall mass and moment of inertia and, by so doing, increases the vibration damping. Refer to appendix G for a more elaborate explanation.

7.6 Testing of the RGPS and Results

7.6.1 Experiment

An experiment was done to determine the accuracy of RGPS using two of uBlox's RCB-4H GPS receivers. The two receivers were placed 2 m apart and each receiver's Dynamic Platform Model was chosen that best describes the behaviour of the platform the GPS receiver will be placed on. As the target will only rotate and not translate, its GPS receiver uses the Stationary Dynamic Platform module. (This model forces the velocities to zero and low pass filters the receiver's position.) It was decided that the Pedestrian Dynamic Platform Model best describes the chaser's behaviour as the hovercraft will have slower accelerations, but in any direction. Both GPS receivers' data was then recorded over a period of about 45 minutes.

7.6.2 Results

Figure 7.15 shows the relative distance between the receivers along the North- and East-axis. From this figure it is clear that this measurement drifts considerably. The relative distance between the 2 GPS receivers are shown in figure 7.16. The mean distance between the 2 GPS receivers is 1.5102 m and the standard deviation is 1.8678 m.

Another, indirect approach would be to integrate the velocities of the chaser to obtain the change in the relative position. For this experiment the integral of the velocities along the North- and East-axis should be zero. The result of integrating the velocities is shown in figures 7.17 and 7.18.

7.6.3 Conclusion

From the results in section 7.6.2 it is clear that the relative distance obtained directly from the GPS receivers is very inaccurate. It was hoped that an accuracy of about 1 meter or less would be obtained. The main reason that this simplistic RGPS configuration does not work is that it is not possible to force the receivers to use the same navigation satellites so that the same disturbances influence both receivers.

²The 30F5011 has 12-bit ADCs and uses a 5 V analog reference.

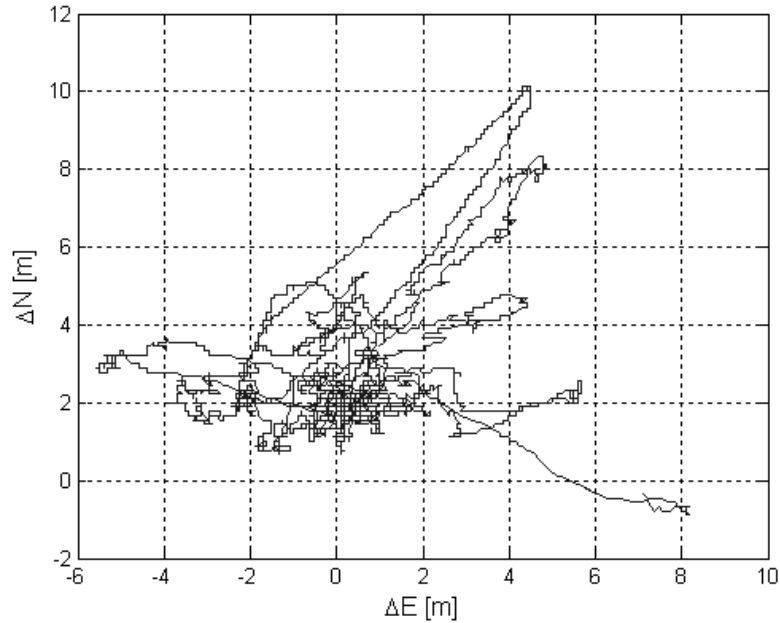


Figure 7.15: Drift of the relative distance between the GPS receivers along the North- and East-axis.

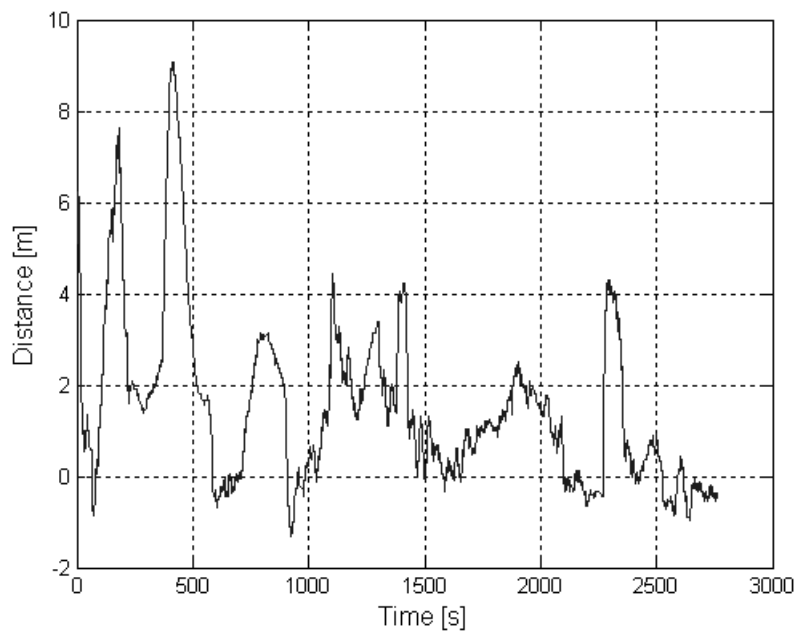


Figure 7.16: Drift of the relative distance between the GPS receivers.

Considering that the test area for the docking sequence would be in the order of 5 by 5 meters (the starting distance between the target and chaser being 2.5 meters) and that the docking sequence is expected to last 2 to 10 minutes, this direct approach will not work for emulation purposes. (If these accuracies were to be obtained in space, it would be more than acceptable for a space application.)

The indirect approach, integrating the velocities, is much more accurate with the error in the relative position remaining within a circle with a radius of 0.55 meters. Apart from the fact

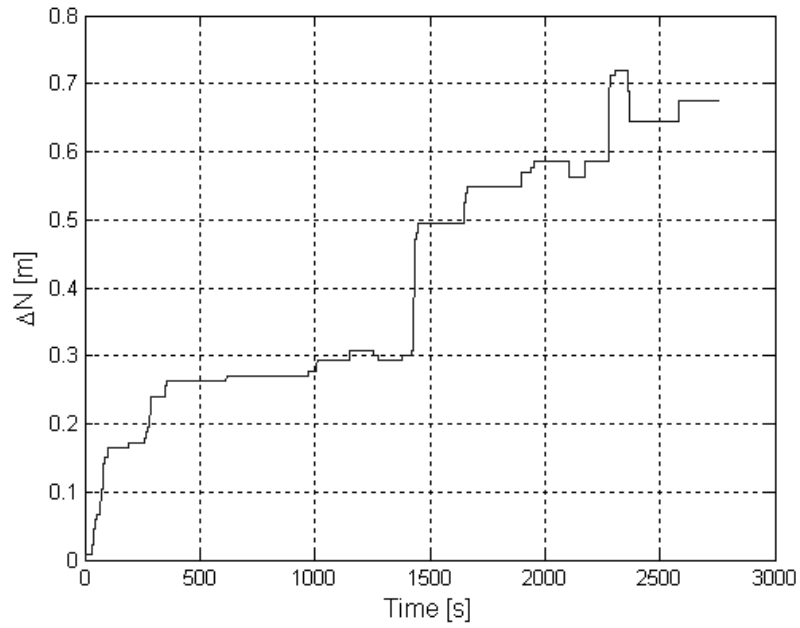


Figure 7.17: The integral of the velocity along the North-axis.

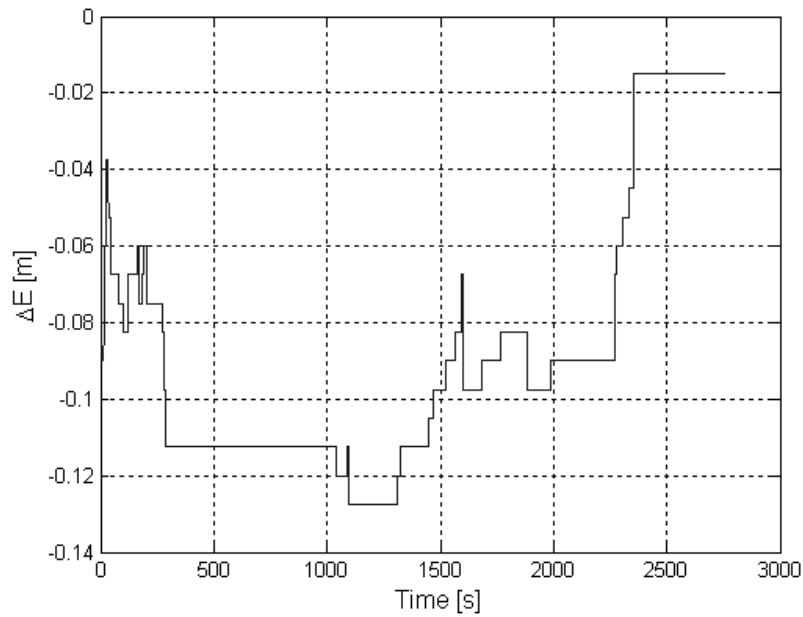


Figure 7.18: The integral of the velocity along the East-axis.

that one does not know how reliable this approach is, the approach does not illustrate RGPS at all. One could just as well only use a single GPS receiver.

Taking all of the above mentioned into account, it was decided to discard the RGPS portion of this thesis.

Chapter 8

The Camera Sensor

As the camera is the most important sensor in this thesis, this whole chapter is dedicated to it. The camera will be relied on for accuracy and therefore much effort has been done to ensure that the camera sensor performs well.

8.1 Description of the Camera Module and Camera Axis

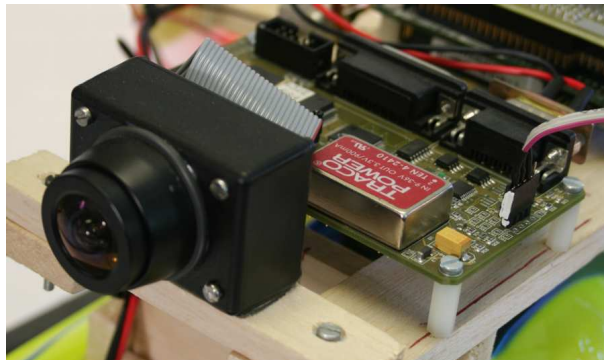


Figure 8.1: The camera unit.

A gray scale camera unit developed originally in the ESL is used in this thesis. The camera unit has an 8051-based micro controller onboard and uses Kodac's KAC-9638 CMOS image sensor. Although the sensor has a maximum resolution of 1288×1032 pixels, only a 1024×1024 window is used. A 4 mm CCTV lens is used resulting in a horizontal and vertical field of view (FOV) of 75° . This wide angle lens is used so that a large part of the target satellite remains in view even when the satellite is in very close proximity of the camera.

Figure 8.2 shows a simplified view of the camera lens and sensor unit's construction. Notice that the lens' theoretical focal point is 8.66 mm behind the lens seat's outer facet. There will be referred to this distance during the camera calibration in section 8.4.

A photo of the sensor is show in figure 8.3 (a) along with a picture of the sensor coordinate system in figure 8.3 (b). The sensor coordinate system's origin coincides with pixel (0,0) and the x-axis runs parallel to the pixel rows of the sensor, while the y-axis runs parallel to the pixel columns of the sensor. Then, the camera coordinate system $(x_{CAM}, y_{CAM}, z_{CAM})$ is defined as

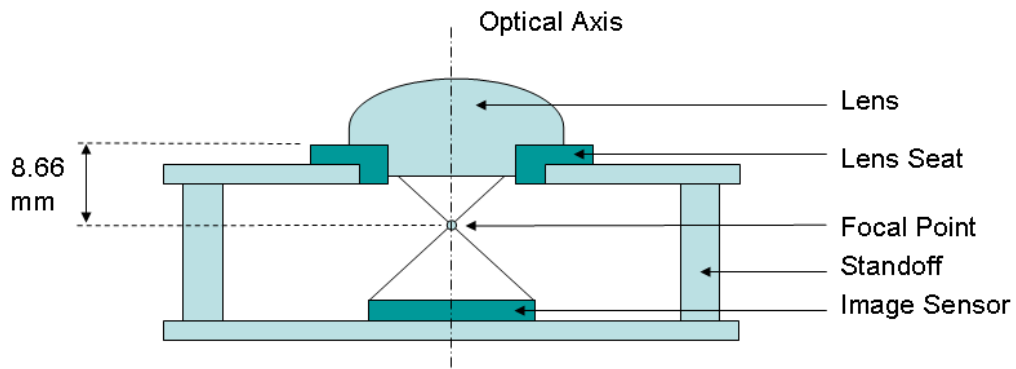


Figure 8.2: Simplified view of the sensor and lens unit's construction.

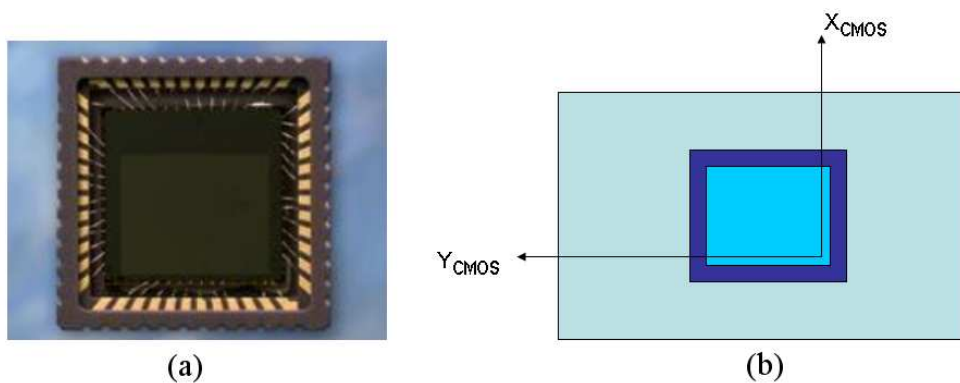


Figure 8.3: (a) The sensor. (b) Sensor axis.

shown in figure 8.4: The z-axis coincides with the optical axis and the x-axis runs parallel to the sensor's x-axis. The y-axis runs parallel to the sensor's y-axis and completes the right handed camera coordinate system. (The sensor is mounted inside the sensor unit in such a way that the sensor's axes, and therefore also the camera's axes, runs parallel to the sides of the unit.) Lastly, the camera axis' origin coincides with the theoretical focal point, as shown in figure 8.2.



Figure 8.4: (a) The camera sensor unit. (b) The camera axis.

8.2 Camera Clamp

The camera clamp, shown in figure 8.5, was designed to accommodate lenses with an outer diameter of 25-40 mm so that it could be used with a variety of lenses.

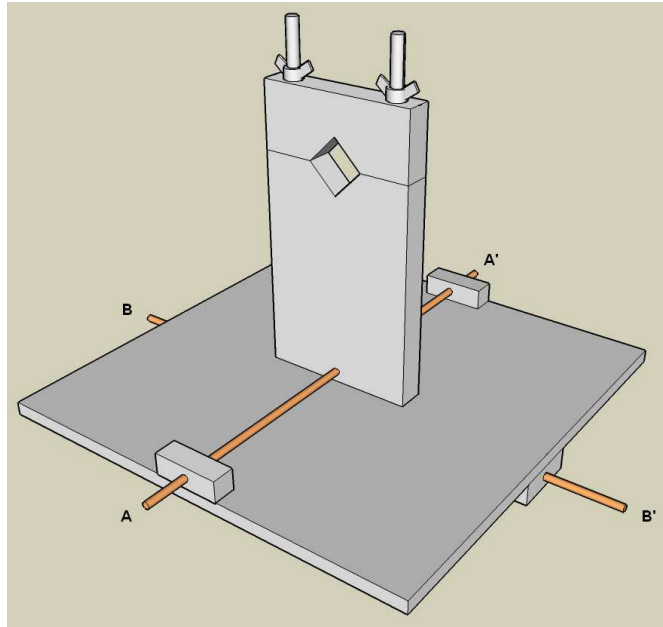


Figure 8.5: The camera clamp.

Two thin pipes are mounted on the clamp to aid the alignment process. Pipe AA' is directly below the centre of the clamp and runs parallel to the inserted lens' ideal optical axis, and pipe BB' runs through the middle of the clamp. The clamp can then be mounted directly on a normal camera tripod.

8.3 Lens Distortion

Most inexpensive camera lenses suffer from clearly visible distortion. Although the total distortion is the result of several distortion sources, the total distortion is dominated by radial distortion [22] [24] [28]. The effect radial distortion has on the geometry of images is categorised as being either "barrel distortion" or "pincushion distortion" as illustrated in figure 8.6. Wide-angle lenses are more prone to suffer from barrel distortion, while telephoto lenses are more susceptible to pincushion distortion [24].

8.3.1 Modeling radial distortion

Radial distortion is an inherent property of lenses and has to be compensated for either optically or mathematically [24]. Optical compensation methods results in a more complex lens system that in turn implies a more expensive lens. Different approaches are used to remove the distortion mathematically. Farid *et al.* [5] is able to remove radial distortion, without any knowledge of the camera's parameters, by exploiting frequency domain characteristics of radial distortion. The disadvantage of their method is that it requires the Fourier transform of

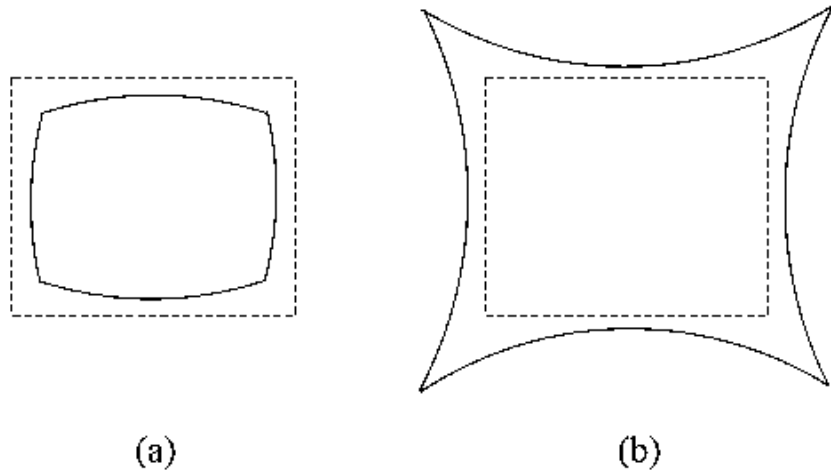


Figure 8.6: Radial Distortion. The dashed line represents the rectangular object as it would appear in the absence of radial distortion and solid line shows the object shape in the presence of (a) barrel and (b) pincushion distortion. Figure taken from [24].

each image, which is computationally expensive. The most common method to compensate for the radial distortion, using the mathematical approach, is to model the distortion. There are several models of which the Polynomial Model is the most popular. This model is defined as follows:

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} x_{COD} \\ y_{COD} \end{bmatrix} + \begin{bmatrix} x_d - x_{COD} \\ y_d - y_{COD} \end{bmatrix} L(r_d, \mathbf{k})$$

Where the distorted radius

$$r_d = \sqrt{(x_d - x_{COD})^2 + (y_d - y_{COD})^2}$$

and the lens mapping function

$$L(r_d, \mathbf{k}) = 1 + k_1 r_d^2 + k_2 r_d^4 + \dots + k_p r_d^{2n}$$

This model transforms the distorted point (x_d, y_d) to the undistorted point (x_u, y_u) . The distorted radius is defined as the distance from the centre of distortion (x_{COD}, y_{COD}) to the distorted point (x_d, y_d) . Although the lens mapping function $L(r_d, \mathbf{k})$ is defined as an infinite power series, Pers [24] states that he and most other authors concluded that for practical purposes a second ($n=1$) or fourth ($n=2$) order polynomial is sufficient. It is also common practice to assume that the centre of distortion coincides with the principle point (u_0, v_0) , resulting in the distortion model used in this thesis:

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} + \begin{bmatrix} x_d - u_0 \\ y_d - v_0 \end{bmatrix} (1 + k_1 r_d^2 + k_2 r_d^4) \quad (8.3.1)$$

With the distorted radius:

$$r_d = \sqrt{(x_d - u_0)^2 + (y_d - v_0)^2} \quad (8.3.2)$$

8.4 Camera Calibration

Camera calibration is the process through which the camera's intrinsic parameters (f, u_0, v_0) and distortion model parameters (k_1, k_2) are determined. Remondino *et al.* [26] gives a good comparison of the different methods used to calibrate a camera. The most common methods are those of Tsai [28] and Zhang [30]. Other interesting methods includes the one of Cucchiara *et al.* [3] where the Hough transform is used to estimate the distortion parameters.

Both Tsai and Zhang's methods were investigated. Tsai's method is aimed at calibrating cameras used for extremely accurate measurements. His method is used for analog cameras and is very complex as he even compensates for uncertainties in the pixel dimensions that arise due to timing errors in the image acquisition hardware. The method proposed by Zhang also has some elaborate equations as he assumes no prior knowledge of the camera. (It is designed to be used by an average person with no knowledge of computer vision.)

Although both these methods are well suited for their individual applications, both are somewhat excessive for the purpose of this thesis. It was therefore decided to use a customised method based on the one of Zhang. The main differences are:

1. The custom method utilises prior knowledge of the camera in the initial estimate of the camera parameters. For example, the lens data sheets specify that the focal length is 4 mm with a tolerance of 5%.
2. In the custom method, the translation and rotation of the calibration object with respect to the camera axis will be measured.
3. Zhang's method requires at least three photos of the test object, each at a different orientation, as his method also estimates the translation and rotation of the test object with respect to the camera axis. As the translation and rotations are measured in the custom method, one photo of the test object is sufficient.

8.4.1 Summary of the calibration procedure

The customised calibration procedure consists of the following steps:

1. Create the planar test pattern. (Note that the terms "test pattern" , "test grid" and "test object" are used interchangeably.)
2. Mount the camera orthogonally in the camera clamp.
3. Place the camera in front of the test pattern so that the test pattern fills the whole image.
4. Rotate the camera clamp so that the camera's optical axis is perpendicular to the test image.
5. Measure the translation and rotation of the test pattern in the camera axis.
6. Take a full resolution picture.
7. Extract the centroids in the image.
8. Use a non-linear minimisation procedure to determine the camera's parameters.

8.5 The Calibration Procedure in Detail

A more elaborate explanation of the camera calibration procedure will be given in this section by showing how the camera used in this thesis was calibrated.

Step 1: A test grid of 11 by 11 points, spaced 100 mm apart, was drawn on a wooden board. The test pattern's 2D axis (x_{OBJ} , y_{OBJ}) origin is placed on the left upper point and is defined as shown in figure 8.7.

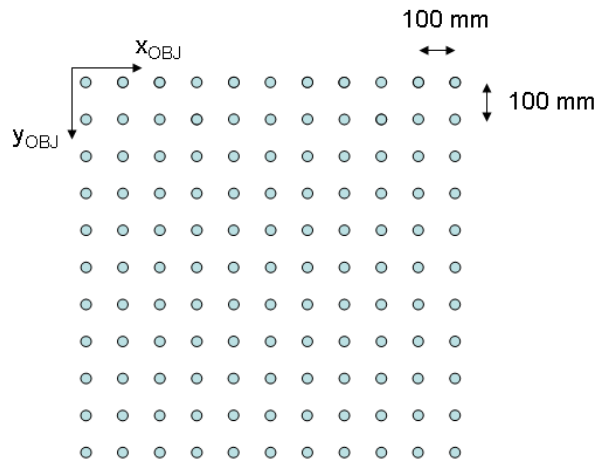


Figure 8.7: The test grid used.

Step 2: Mount the camera orthogonally in the camera clamp. This is achieved by ensuring that the distance between the clamp's base and the camera unit is identical on both sides of the clamp as illustrated in figure 8.8.

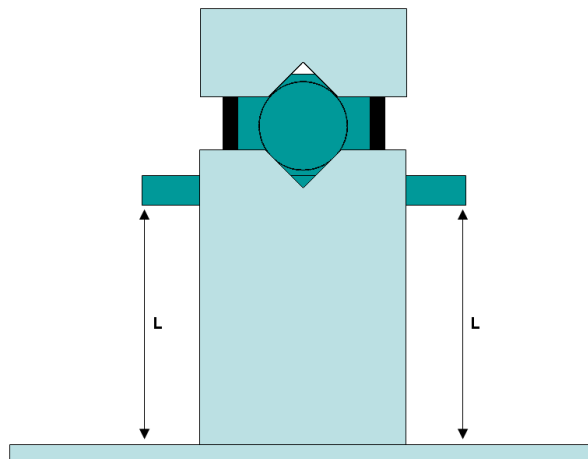


Figure 8.8: Mounting the camera in the clamp.

Step 3: Next, the camera clamp is mounted on a sturdy tripod and this camera setup is placed in front of the test pattern. At this stage one has to take two things into consideration. Firstly, the camera must be placed close enough to the test pattern so that it fills the whole image.

This is done so that the radial distortion parameters can be estimated more accurately as the distortion is more prominent closer to the edges of the image. Secondly, one wants to place the camera as far away from the test pattern as possible so that the distance measurement errors are smaller, percentage wise. For example: If the distance between the camera and the test pattern along the camera's z-axis is Z_{OBJ}^{CAM} and the measurement error is ϵ_Z , then the percentage error is given by:

$$e = 100 \left(\frac{\epsilon_Z}{Z_{OBJ}^{CAM}} \right) \% \quad (8.5.1)$$

From equation 8.5.1, taking into account that the measurement error should remain fairly constant, it is clear why one would want to maximise the distance between the test object and the camera. Also, the camera's theoretical focal point was used in determining the location of camera axis' origin. The error in this assumption is lumped together with the measurement error ϵ_Z . Another advantage of maximising the distance is that it allows the angles between the camera and test pattern to be measured more accurately. Finally, a compromise has to be made where the distance between the test pattern and camera is determined by the size of the test pattern.

Step 4: Rotate the camera clamp so that the camera's axis is orthogonal to the test pattern's axis as illustrated in figure 8.9.

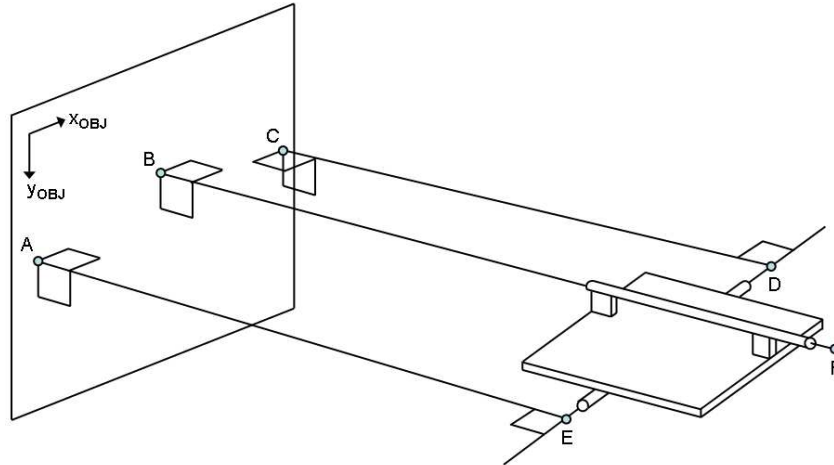


Figure 8.9: Mount the camera axis orthogonally with respect to the test pattern's axis.

Step 5: Measure the translation and rotation of the test pattern's axis in the camera axis. From figure 8.10 it can be seen that the orientation of the test object in the camera axis is:

$$\left[\phi, \theta, \psi \right]_{CAM \rightarrow OBJ} = \left[0^\circ, 0^\circ, 90^\circ \right]$$

using the Euler 3-2-1 convention.

The translation along the camera's z-axis is measured fairly directly. The distance between the test pattern and the centre of the the pipe running through the middle of the camera clamp

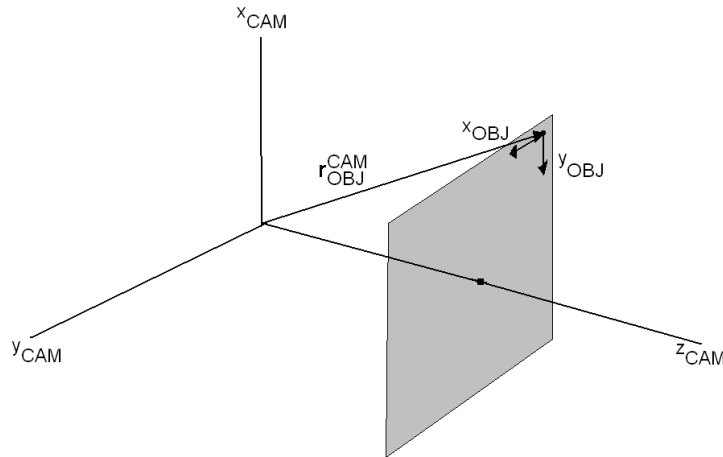


Figure 8.10: The test object's axis orientation in the camera axis.

(\overline{AE} in figure 8.9) was measured to be 588 mm. To this was added half of the clamps width (6 mm) and distance of the theoretical focal point behind the lens seat's outer facet (8.66 mm). This gives $Z_{OBJ}^{CAM} = 602.66$ mm.

An indirect approach was followed to calculate the other two distances by aiming a laser pointer through the pipe running parallel to the camera's optical axis and marking its position in the test object's axis. The position of this mark was found to be 427 mm and 588 mm along x_{OBJ} and y_{OBJ} , respectively. Then by taking into account the orientation of the test object's axis with respect to the camera axis, and the fact that the centre of the pipe is 104 mm below the camera's ideal optical axis (centre of the lens barrel), it was determined that $X_{OBJ}^{CAM} = 484$ mm and $Y_{OBJ}^{CAM} = -427$ mm. Note that it is assumed that the camera's optical axis runs through the middle of the lens barrel. The error in this assumption only translates to an offset in the principle point.

Step 6: Take a full resolution picture. This picture is shown in figure 8.11. Note the camera setup's shadow in the middle, left of the picture. Also, it can clearly be seen that this wide angle lens causes barrel distortion.

Step 7: Extract the centroids in the image. Due to the nonuniform lighting conditions it is not possible to simply use one global threshold value to extract the centroids. Therefore a slightly longer process has to be followed. First the centroid locations are isolated using an image manipulation program, such as GIMP, resulting in the image shown in figure 8.12 (a). Then a MATLAB program was used to semi-automatically threshold each centroid location (Local thresholding). The program is semi-automatic in the sense that one has to indicate the location of the centroids by clicking in the vicinity of the centroid. The output of this process is shown in figure 8.12 (b). Next the background is removed resulting in the image shown in figure 8.13. The last step is to uniquely identify each centroid. (The reason for this will become clear in the next step.) This was also done using a MATLAB program. All of the centroids are stored in an array in the same order as one clicked on them. Therefore one only has to click on the centroids in the correct order.

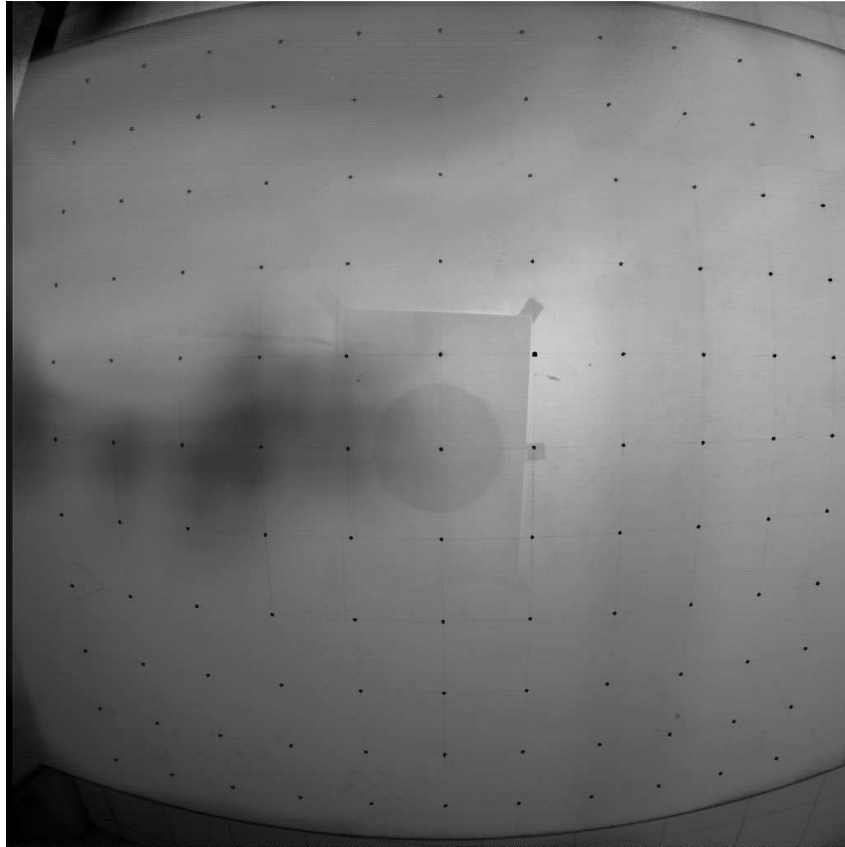


Figure 8.11: The photo used for calibration.

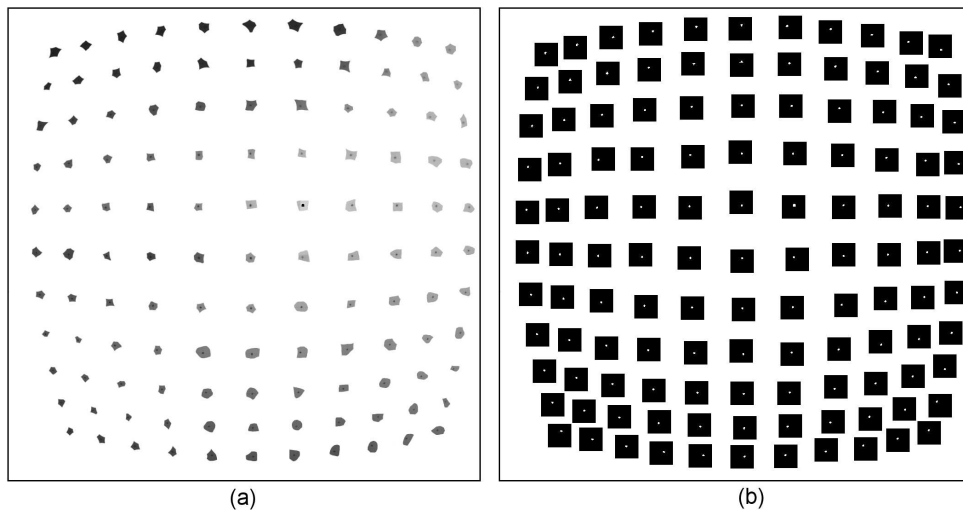


Figure 8.12: (a) Isolating the centroids locations. (b) The result of local thresholding.

Step 8: Now the camera's parameters are determined using a non-linear minimisation procedure. The basic idea is to minimise the error between the ideal centroids obtained by projecting the test grid's points onto the camera's image plane, and the ideal centroids obtained by correcting the distorted centroids from the picture using equations 8.3.1 and 8.3.2. This is illustrated in figure 8.14. If $\mathbf{M}_j = [x_{tp_j}, y_{tp_j}, 0]^T$ is the position of a point j in the test pattern's

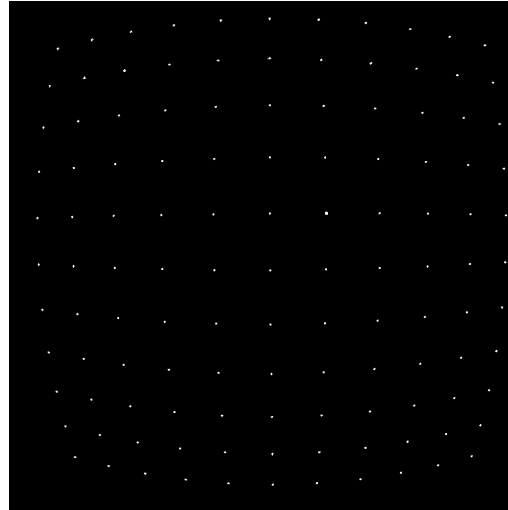


Figure 8.13: The final processed image containing the centroids.

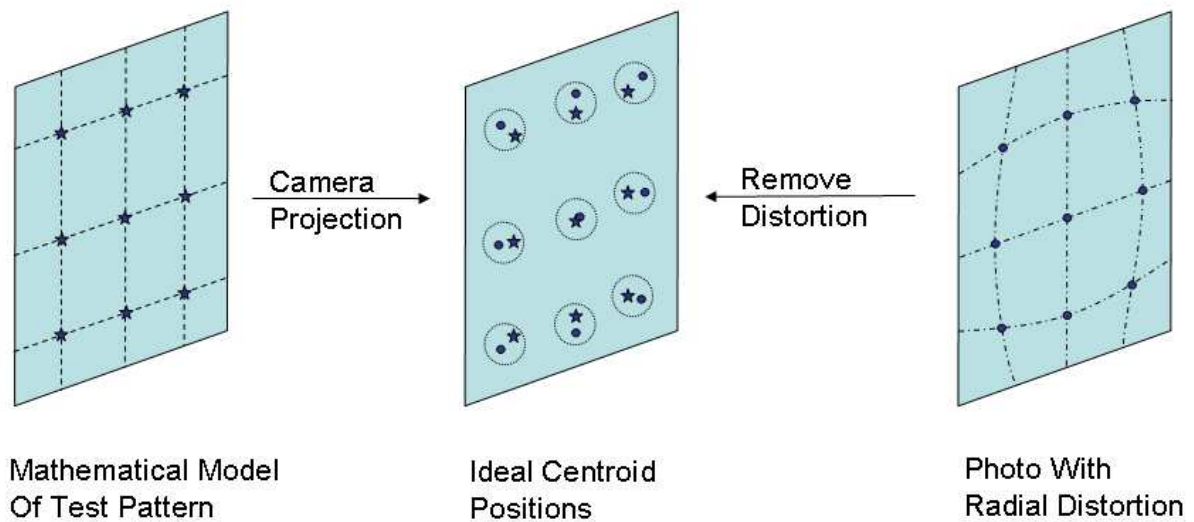


Figure 8.14: The basic concept used to determine the camera's parameters.

axis, then its projection, \mathbf{m}_j , onto the camera's image plane, using the projective camera model, is described by the matrix equation¹:

$$\tilde{\mathbf{m}}_j = \begin{bmatrix} \frac{f}{p_x} & 0 & u_0 \\ 0 & \frac{f}{p_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\mathbf{T}_{OBJ \rightarrow CAM} \quad \mathbf{r}_{OBJ}^{CAM} \right] \tilde{\mathbf{M}}_j \quad (8.5.2)$$

Where:

- The homogeneous vector, $\tilde{\mathbf{m}}_j = [\tilde{x}_j, \tilde{y}_j, \tilde{z}_j]^T$, gives the position of the projected test pattern's point j in the image plane.
- f is the focal length in meters.

¹For an elaborate explanation on how to use this camera matrix and homogeneous vectors, refer to appendix E.

- Each pixel's size is p_X by p_Y meters. For the camera used in this thesis $p_X = p_Y = 6 \times 10^{-6}$ m.
- (u_0, v_0) is the principle point. (Where the optical axis intersect with the image plane.)
- $\mathbf{T}_{OBJ \rightarrow CAM}$ is the DCM that transforms a vector from the test object axis to the camera axis. It is determined using the information from step 5.
- \mathbf{r}_{OBJ}^{CAM} is the position of the test object axis origin in the camera axis. From step 5 it is given as $\mathbf{r}_{OBJ}^{CAM} = [0.484, -0.427, 0.602]^T$.
- $\tilde{\mathbf{M}}_j = \begin{bmatrix} \mathbf{M}_j \\ 1 \end{bmatrix} = [x_{tp_j}, y_{tp_j}, 0, 1]^T$ is the homogeneous vector that gives the position of the point j in the test object axis.

This equation yields:

$$x_{u1_j} = \frac{\tilde{x}_j}{\tilde{z}_j}$$

$$y_{u1_j} = \frac{\tilde{y}_j}{\tilde{z}_j}$$

Where (x_{u1_j}, y_{u1_j}) is the ideal centroid obtained by projecting a point j from the test grid onto the image plane. Also note that (x_{u1_j}, y_{u1_j}) is a function of the unknown parameters f , u_0 and v_0 .

Each distorted centroid (x_{d_j}, y_{d_j}) from the picture is transformed to the corrected, ideal centroid (x_{u2_j}, y_{u2_j}) using equations 8.3.1 and 8.3.2. Note that (x_{u2_j}, y_{u2_j}) is a function of k_1 and k_2 . Also, it is important to use the point (x_{d_j}, y_{d_j}) in the picture that corresponds to the point (x_{u1_j}, y_{u1_j}) in the test pattern.

Now the Levenberg-Marquardt Method (Levmar for short) of non-linear minimisation is used to determine the camera's parameters f , u_0 , v_0 , k_1 and k_2 using the cost function:

$$F = \sum_{j=0}^{N-1} \left[(x_{u1_j} - x_{u2_j})^2 + (y_{u1_j} - y_{u2_j})^2 \right] \quad (8.5.3)$$

Where N is the amount of centroids. The initial values for the parameters are $f = 0.004$ m (from the lens' data sheet), $u_0 = v_0 = 512$ (expected values) and $k_1 = k_2 = 0$ (standard initial values).

For a detailed description of the Levmar algorithm, and other non-linear minimisation procedures, refer to the document from Madsen *et al.*[19]. Lourakis [16] also provides a good description of the Levmar algorithm.

8.6 Results from the Calibration Procedure

8.6.1 Results

Table 8.1 summarises the results that were obtained from the calibration procedure:

Parameter	Value
f	4.0971 mm
u_0	541.0234 pixels
v_0	456.0833 pixels
k_1	6.0227×10^{-7}
k_2	1.3581×10^{-12}
RMS Error	1.3288 pixels

Table 8.1: Calibration Results.

8.6.2 Evaluation of the results

All the results are acceptable. The lens' data sheet states that the lens' focal length is 4 mm with a tolerance of $\pm 5\%$ so that the estimated value of 4.0971 mm is well within this range.

Under ideal circumstances the principle point should be $(u_0, v_0) = (512, 512)$. The estimated principle point implies that the sensor's position is out by 0.174 mm along the one axis and by 0.336 mm along the other axis, which is possible.

Lastly, the estimated values of k_1 and k_2 is also acceptable as they are such that the undistorted radius $r_u = r_d(1 + k_1 r_d^2 + k_2 r_d^4) \geq r_d$ as is true for barrel distortion.

8.6.3 Visual test

Another way to test the results is visually. By inverting the radial distortion model, numerically, it is possible to create a mathematical photo of the test pattern. This mathematical photo is then superimposed on the original photo as show in figure 8.15. From this figure it can be seen that the model is very accurate as the white points of the mathematical photo basically lies on top of the darker points of the test pattern (Actual photo).

Next, the original photo (repeated in figure 8.17 for convenience) is show together with the corrected image in figure 8.18. It should be clear that there is an immense improvement.

Lastly, figure 8.16 shows the source pixels from the original image used during the correction process. From this picture it can be seen that barrel distortion compresses the image, so that the lens, suffering from barrel distortion, effectively see more than an ideal lens with no distortion.

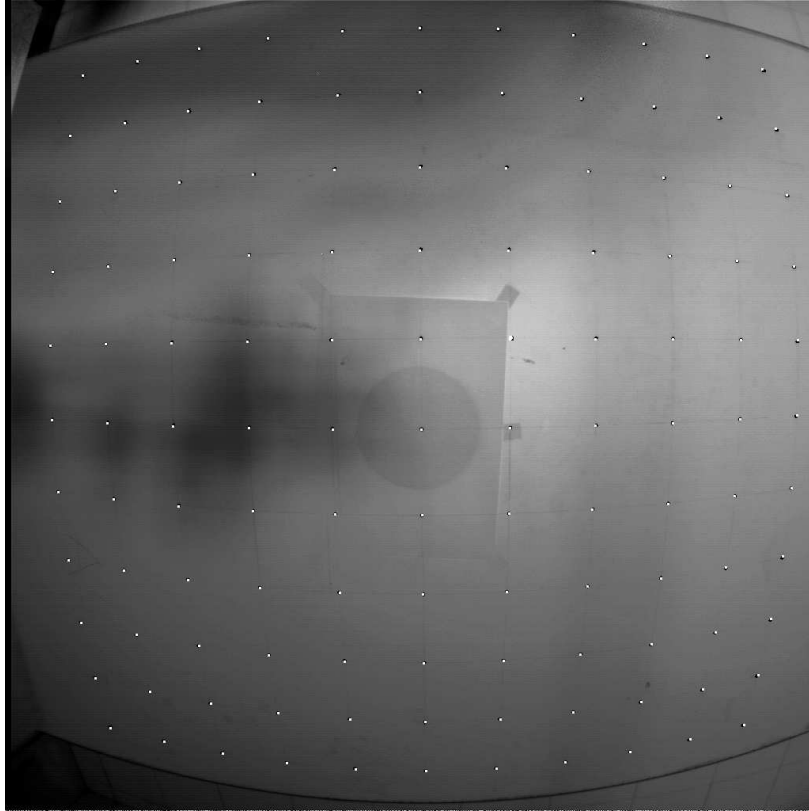


Figure 8.15: The mathematical photo (white points) superimposed on the actual photo (black spots).

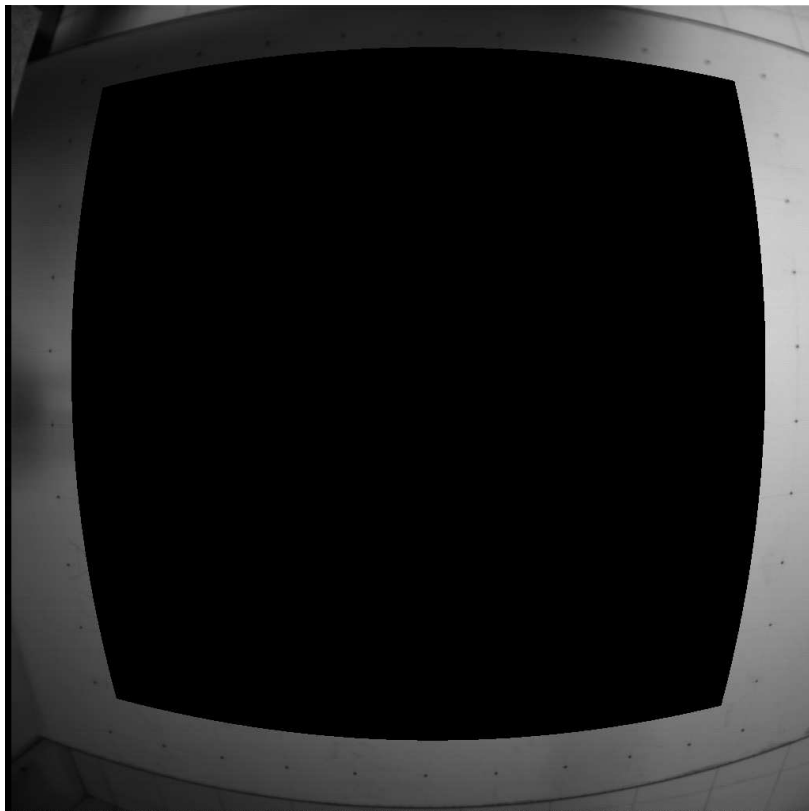


Figure 8.16: The pixels used during the correction process. (The black area in the centre of the image.)

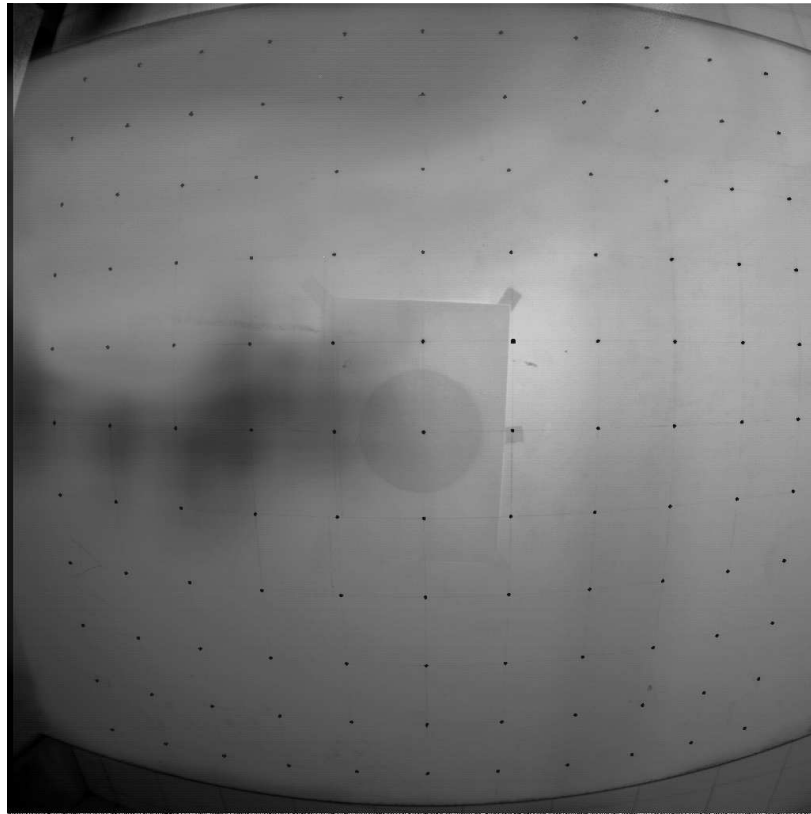


Figure 8.17: The original photo.

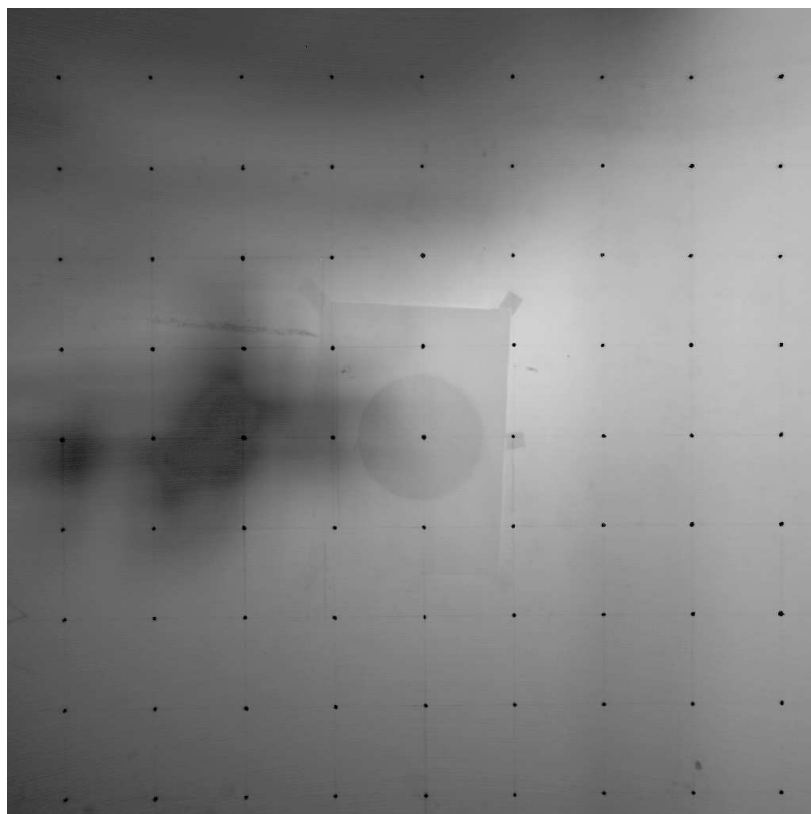


Figure 8.18: The corrected picture.

Chapter 9

Tailoring DF Malan's EKF

In this chapter DF Malan's filter is tailored for the 2D emulation.

9.1 Preface

The filter designed by Malan [20] is used to estimate the orientation and translation of a target satellite in a stationary camera axis by using the target's lights' centroids. His filter has 13 states:

$$\bar{\mathbf{x}} = [\bar{x} \quad \bar{y} \quad \bar{z} \quad \bar{v}_X \quad \bar{v}_Y \quad \bar{v}_Z \quad \bar{q}_1 \quad \bar{q}_2 \quad \bar{q}_3 \quad \bar{q}_4 \quad \bar{\omega}_{T_X} \quad \bar{\omega}_{T_Y} \quad \bar{\omega}_{T_Z}]^T$$

Where:

- $\bar{\mathbf{r}}_{TGT}^{CAM} = [\bar{x} \quad \bar{y} \quad \bar{z}]^T$ is the estimated position of the target satellite's body axis in the camera axis.
- $\bar{\mathbf{v}}_{TGT}^{CAM} = [\bar{v}_X \quad \bar{v}_Y \quad \bar{v}_Z]^T$ is the estimated velocity of the target satellite's body axis in the camera axis.
- The quaternions $[\bar{q}_1 \quad \bar{q}_2 \quad \bar{q}_3 \quad \bar{q}_4]^T$ give the estimated rotation of the target satellite's body axis with respect to the camera axis, and
- The estimated angular velocities of the target satellite in its own body axis is given by $[\bar{\omega}_{T_X} \quad \bar{\omega}_{T_Y} \quad \bar{\omega}_{T_Z}]^T$.

As the emulation will only be in 2 dimensions, many of these states are unnecessary. From figure 9.1 can be seen that the target's position along the camera's x-axis, \bar{x} , is a constant that does not need to be estimated (It must, however, still be measured manually as some equations still require it.) Due to the movement in a 2D plane, the velocity, \bar{v}_X , and angular velocities, $\bar{\omega}_{T_X}$, and, $\bar{\omega}_{T_Y}$, are also zero. Then the quaternions can be replaced by an single angle, $\bar{\psi}$, as will be shown in section 9.2.

As stated before, Malan's filter assumes a stationary camera. In this thesis, the camera axis rotates with the chaser axis and therefore the orientation of the target's body axis with respect to the camera axis is influenced by both the chaser's and target's angular velocities. Because of this, the remaining angular rate, $\bar{\omega}_{T_Z}$, becomes $\bar{\omega}_{T_Z/C_Z}$.

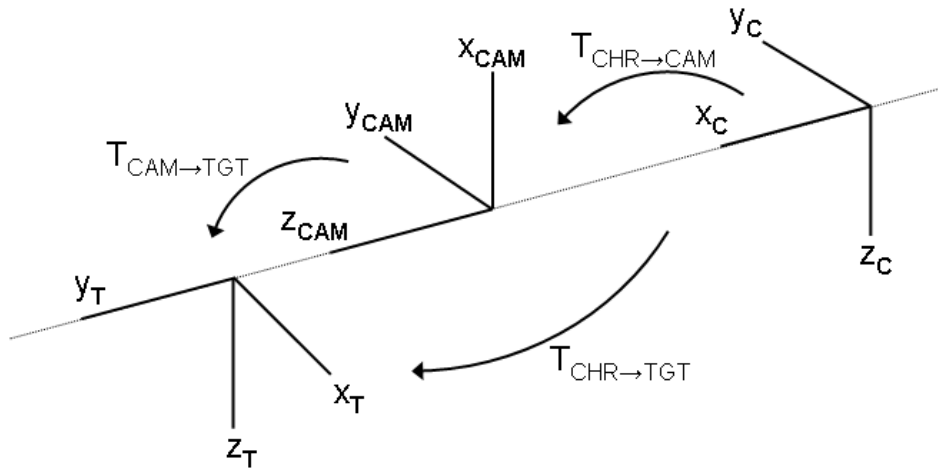


Figure 9.1: All the relevant axes.

Therefore, this thesis only requires the following 6 states:

$$\bar{\mathbf{x}}_{DF} = [\bar{z} \quad \bar{y} \quad \bar{v}_Z \quad \bar{v}_Y \quad \bar{\psi} \quad \bar{\omega}_{Tz/Cz}]^T$$

This, in turn, implies a simpler, faster Extended Kalman Filter that is less processor intensive. The objective of the rest of this chapter is to describe these simplifications.

9.2 Using Euler Angles

9.2.1 Using Euler 1-2-3 instead of Euler 3-2-1

Referring to figure 9.1, note that the rotation from the chaser's body axis to the target's body axis can be fully described by a single rotation, ψ , around the chaser's z -axis. The goal is to directly estimate this yaw angle, ψ , and the first step is to use Euler angles instead of quaternions. As there is no unique definition for Euler angles, the ESL has standardised on Euler 3-2-1. (First yaw, then pitch and then roll. Stated differently: First rotate about the z -axis, then the y -axis and lastly about the x -axis.) The problem with this convention is that the rotation, ψ , cannot be determined directly as explained in the next paragraph.

A general problem with Euler angles is that it is typically difficult to combine two rotations. Given that one has the Euler angles $[\phi, \theta, \psi]_{CHR \rightarrow CAM}$ (Chaser Body Axis to Camera Axis) and $[\phi, \theta, \psi]_{CAM \rightarrow TGT}$ (Camera Axis to Target Body Axis), then $[\phi, \theta, \psi]_{CHR \rightarrow TGT}$ (Chaser Body Axis to Target Body Axis) cannot be calculated in a straightforward way.

Typically one first have to determine the DCMs $\mathbf{T}_{CHR \rightarrow CAM}$, which is a constant matrix, and $\mathbf{T}_{CAM \rightarrow TGT}$, determined by Malan's EKF. Then $\mathbf{T}_{CHR \rightarrow TGT} = \mathbf{T}_{CAM \rightarrow TGT} \mathbf{T}_{CHR \rightarrow CAM}$ must be calculated. From $\mathbf{T}_{CHR \rightarrow TGT}$ one then determines $[\phi, \theta, \psi]_{CHR \rightarrow TGT}$. But this is a very tedious approach. Further investigations proved that a simpler solution is to just use a different order of rotations.

First some background on Euler angles. The first two rotations gives the orientation of the Euler axis, $\hat{\mathbf{e}}$, and the last rotation then gives the rotation, θ_e , about the Euler axis. This is illustrated in figure 9.2. For example: In Euler 3-2-1, after the target's body axis has been

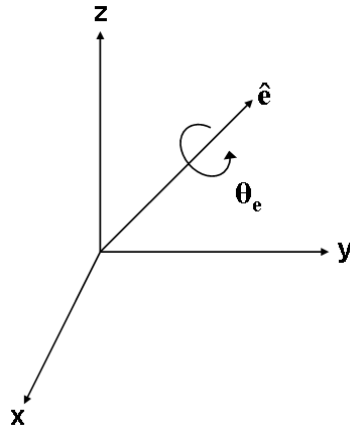


Figure 9.2: The Euler axis, $\hat{\mathbf{e}}$, and Euler angle, θ_e . (Picture adapted from Wikipedia)

firstly rotated about the z-axis and then the y-axis, the rotated x-axis forms the Euler axis. The rotation about this rotated x-axis is then the Euler angle.

Taking the above mentioned into account, it was decided to use Euler 1-2-3 (First rotate around the x-axis, then the y-axis and lastly about the z-axis) Then, in the DCM that gives the orientation of the target's body axis with respect to the camera axis, the first two rotations aligns the Euler axis with the target's z-axis. (Note that these two rotations will be constants¹) The last rotation, the Euler angle, is the rotation around target's z-axis which is the angle of interest. Therefore, if Euler 1-2-3 is used to represent the DCM from the camera axis to the target's body axis, one can directly determine the rotation, ψ , from the chaser's body axis to the target's body axis.

In order to fully describe the target's orientation with respect to the chaser using only one angle, ψ , and to simplify the maths, the camera axis is mounted at right angles with respect to the chaser's body axis, so that:

$$[\phi, \theta, \psi]_{CAM \rightarrow TGT} = [0^\circ, -90^\circ, \psi]$$

9.2.2 The Euler 1-2-3 DCM

The Euler 1-2-3 DCM is determined by multiplying out the following rotation matrices in the correct order.

The rotation around the z-axis:

$$\mathbf{T}_Z = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

¹When the air cushion is deflated. Any tilting of the air cushion is neglected.

The rotation around the y-axis:

$$\mathbf{S}_Y = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

The rotation around the x-axis:

$$\mathbf{R}_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

The forward DCM, using Euler 1-2-3, is then:

$$\begin{aligned} \mathbf{T} &= \mathbf{T}_Z \mathbf{S}_Y \mathbf{R}_X \\ &= \begin{bmatrix} \cos(\psi) \cos(\theta) & \sin(\psi) \cos(\phi) + \cos(\psi) \sin(\theta) \sin(\phi) & \sin(\psi) \sin(\phi) - \cos(\psi) \sin(\theta) \cos(\phi) \\ -\sin(\psi) \cos(\theta) & \cos(\psi) \cos(\phi) - \sin(\psi) \sin(\theta) \sin(\phi) & \cos(\psi) \sin(\phi) + \sin(\psi) \sin(\theta) \cos(\phi) \\ \sin(\theta) & -\cos(\theta) \sin(\phi) & \cos(\theta) \cos(\phi) \end{bmatrix} \end{aligned}$$

The inverse DCM is then:

$$\begin{aligned} \mathbf{T}^{-1} &= \mathbf{T}^T \\ &= \begin{bmatrix} \cos(\psi) \cos(\theta) & -\sin(\psi) \cos(\theta) & \sin(\theta) \\ \sin(\psi) \cos(\phi) + \cos(\psi) \sin(\theta) \sin(\phi) & \cos(\psi) \cos(\phi) - \sin(\psi) \sin(\theta) \sin(\phi) & -\cos(\theta) \sin(\phi) \\ \sin(\psi) \sin(\phi) - \cos(\psi) \sin(\theta) \cos(\phi) & \cos(\psi) \sin(\phi) + \sin(\psi) \sin(\theta) \cos(\phi) & \cos(\theta) \cos(\phi) \end{bmatrix} \end{aligned}$$

With this background, one is now in the position to do the simplifications.

9.3 The Projective Camera Model in Matrix Form

In this section the equations are derived that are required to determine the estimated positions of the target's lights' centroids in the image. Start with the projective camera model in matrix² form, as it would be given in a textbook [10]:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} \frac{f}{p_X} & 0 & u_0 \\ 0 & \frac{f}{p_Y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix}$$

Where the homogeneous vector $[\tilde{x}, \tilde{y}, \tilde{z}]^T$ is the output of the camera, \mathbf{R} is the inverse DCM that converts vectors from the observed object's body axis to the camera axis. The translation \mathbf{t} is the position of the object's body axis in the camera axis. $[a, b, c]^T$ is the location of a point in the object's body axis. (u_0, v_0) is the optical centre, f is the focal length, p_X and p_Y is the size of a pixel.

²For an elaborate explanation on how to use this camera matrix and homogeneous vectors, refer to appendix E.

In this application, the observed object is the target satellite and the point in the observed object's axis is the target's lights. Using the notation introduced in section 4.1, but using the estimated states, this equation can be rewritten as:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} \frac{f}{p_x} & 0 & u_0 \\ 0 & \frac{f}{p_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\bar{\mathbf{T}}_{TGT \rightarrow CAM} \quad \bar{\mathbf{r}}_{TGT}^{CAM} \right] \begin{bmatrix} \mathbf{r}_{L_n}^{TGT} \\ 1 \end{bmatrix} \quad (9.3.1)$$

Remembering that $[\phi, \theta, \psi]_{CAM \rightarrow TGT} = [0^\circ, -90^\circ, \psi]$, but using the estimated angle, the inverse DCM becomes:

$$\bar{\mathbf{T}}_{TGT \rightarrow CAM} = \begin{bmatrix} 0 & 0 & -1 \\ \sin(\bar{\psi}) & \cos(\bar{\psi}) & 0 \\ \cos(\bar{\psi}) & -\sin(\bar{\psi}) & 0 \end{bmatrix}$$

The pixels of the camera are square so that:

$$p = p_x = p_y$$

Define:

- $\bar{\mathbf{r}}_{TGT}^{CAM} = [X, \bar{y}, \bar{z}]^T$
- $\mathbf{r}_{L_n}^{TGT} = [a, b, c]^T$
- $\kappa = \frac{f}{p}$

Note that X is a constant as it does not need to be estimated in this 2D application as was motivated in section 9.1. Then the estimated position of a light in the image³ is determined by solving equation 9.3.1:

$$\begin{aligned} \bar{x}_{sensor} &= \frac{\tilde{x}}{\tilde{z}} \\ &= \frac{\kappa(X - c)}{\cos(\bar{\psi})a - \sin(\bar{\psi})b + \bar{z}} + u_0 \end{aligned} \quad (9.3.2)$$

$$\begin{aligned} \bar{y}_{sensor} &= \frac{\tilde{y}}{\tilde{z}} \\ &= \frac{\kappa(a \sin(\bar{\psi}) + b \cos(\bar{\psi}) + \bar{y})}{\cos(\bar{\psi})a - \sin(\bar{\psi})b + \bar{z}} + v_0 \end{aligned} \quad (9.3.3)$$

³Note that these equations can also be determined solving equations 4.4.1 and 4.4.2.

9.4 The Customised Filter

9.4.1 The state vector

The state vector is defined as shown in section 9.1:

$$\bar{\mathbf{x}}_{DF} = [\bar{z} \quad \bar{y} \quad \bar{v}_Z \quad \bar{v}_Y \quad \bar{\psi} \quad \bar{\omega}_{Tz/Cz}]^T$$

Where:

- \bar{z} and \bar{y} is the estimated position of the target's body axis along the camera's z- and y-axis, respectively.
- \bar{v}_Z and \bar{v}_Y is the estimated velocity of the target's body axis along the camera's z- and y-axis, respectively.
- The estimated rotation of the target's body axis with respect to the chaser's body axis is given by $\bar{\psi}$.
- The estimated rate at which the rotation of the target's body axis with respect to the chaser's body axis, $\bar{\psi}$, changes is given by $\bar{\omega}_{Tz/Cz} = \bar{\omega}_{Tz} - \bar{\omega}_{Cz}$.

9.4.2 Propagation

The propagation model is given by the continuous differential equation:

$$\dot{\bar{\mathbf{x}}}_{DF} = \mathbf{f}(\bar{\mathbf{x}}_{DF}) + \mathbf{G}_1 \mathbf{w}_1$$

Where:

$$\mathbf{f}(\bar{\mathbf{x}}_{DF}) = \begin{bmatrix} \bar{v}_Z \\ \bar{v}_Y \\ 0 \\ 0 \\ \bar{\omega}_z \\ 0 \end{bmatrix}$$

The above equation is fully linear and can therefore be written as:

$$\mathbf{f}(\bar{\mathbf{x}}_{DF}) = \mathbf{F}_k \bar{\mathbf{x}}_{DF}$$

With:

$$\mathbf{F}_k = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

But, \mathbf{F}_k is still continuous and has to be transformed to its discrete equivalent:

$$\begin{aligned}\mathbf{F}_{dk} &= \mathbf{e}^{\mathbf{F}_k T_{s_0}} \\ &= \mathbf{I}_{6 \times 6} + \mathbf{F}_k T_{s_0} + \frac{\mathbf{F}_k^2 T_{s_0}^2}{2!} + \frac{\mathbf{F}_k^3 T_{s_0}^3}{3!} + \dots \text{TaylorExpansion}\end{aligned}$$

Where T_{s_0} is the outer loop sample period of 100 ms. Due to the form of \mathbf{F}_k , \mathbf{F}_k^n with $n \geq 2$ is $\mathbf{0}_{6 \times 6}$ so that it is not necessary to approximate \mathbf{F}_{dk} :

$$\begin{aligned}\mathbf{F}_{dk} &= \mathbf{I}_{6 \times 6} + \mathbf{F}_k T_{s_0} + \mathbf{0}_{6 \times 6} \\ &= \begin{bmatrix} 1 & 0 & T_{s_0} & 0 & 0 & 0 \\ 0 & 1 & 0 & T_{s_0} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T_{s_0} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\end{aligned}$$

It is assumed that the process noise, \mathbf{w}_1 , takes the form of disturbance accelerations and angular accelerations. From this:

$$\mathbf{G}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

And:

$$\mathbf{w}_1 = \begin{bmatrix} n_Z \\ n_Y \\ n_\omega \end{bmatrix}$$

Where n_Z and n_Y is the disturbance accelerations along the camera's z- and y-axis, respectively. The disturbance angular acceleration is given by n_ω .

9.4.3 Measurement

The measurement is given equations 9.3.2 and 9.3.3:

$$\begin{aligned}\begin{bmatrix} \bar{x}_{sensor} \\ \bar{y}_{sensor} \end{bmatrix} &= \mathbf{h}(\bar{\mathbf{x}}_{DF}) + \mathbf{v}_1 \\ &= \begin{bmatrix} \frac{\kappa(X-c)}{\cos(\bar{\psi})a - \sin(\bar{\psi})b + \bar{z}} + u_0 \\ \frac{\kappa(a \sin(\bar{\psi}) + b \cos(\bar{\psi}) + \bar{y})}{\cos(\bar{\psi})a - \sin(\bar{\psi})b + \bar{z}} + v_0 \end{bmatrix} + \mathbf{v}_1\end{aligned}$$

Where \mathbf{v}_1 is the measurement noise. The Jacobian, \mathbf{H}_k , was determined using MATLAB and is shown here:

$$\mathbf{H}_k(1,:)^T = \begin{bmatrix} \frac{(c-X)\kappa}{(\cos(\bar{\psi})a - \sin(\bar{\psi})b + \bar{z})^2} \\ 0 \\ 0 \\ 0 \\ \frac{(b \cos(\bar{\psi}) + a \sin(\bar{\psi}))\kappa(-c+X)}{(\cos(\bar{\psi})a - \sin(\bar{\psi})b + \bar{z})^2} \\ 0 \end{bmatrix}$$

$$\mathbf{H}_k(2,:)^T = \begin{bmatrix} \frac{(-a \sin(\bar{\psi}) - b \cos(\bar{\psi}) - \bar{y})\kappa}{(\cos(\bar{\psi})a - \sin(\bar{\psi})b + \bar{z})^2} \\ \frac{\kappa}{\cos(\bar{\psi})a - \sin(\bar{\psi})b + \bar{z}} \\ 0 \\ 0 \\ \frac{\kappa((\cos(\bar{\psi}))^2 a^2 + \cos(\bar{\psi})a\bar{z} + (\sin(\bar{\psi}))^2 b^2 - \sin(\bar{\psi})b\bar{z} + a^2(\sin(\bar{\psi}))^2 + b^2(\cos(\bar{\psi}))^2 + \bar{y}b \cos(\bar{\psi}) + \bar{y}a \sin(\bar{\psi}))}{(\cos(\bar{\psi})a - \sin(\bar{\psi})b + \bar{z})^2} \\ 0 \end{bmatrix}$$

9.5 The Noise Variances

9.5.1 Process noise

Due to the isolated implementation of Malan's filter, the disturbance accelerations and disturbance angular acceleration experienced by Malan's filter is the relative accelerations and angular acceleration between the chaser and target. (It is seen by Malan's filter as disturbances as this filter is unaware of any accelerations and angular accelerations commanded.) As the revolving target's position is stationary, the relative acceleration is only the acceleration of the chaser.

The standard deviations of the disturbance acceleration were chosen as follow: Remember that the camera's z-axis is parallel to the chaser's x-axis and that the chaser's and camera's y-axis are parallel to each other. Also remember that there is 2 actuators along the chaser's x-axis. Given that the chaser's mass is 1.75 kg and that the magnitude of the actuator forces along the chaser's x-axis and y-axis is software limited to 0.2 N and 0.3 N, respectively, the standard deviations are chosen as:

$$\sigma_Z = 2(0.2)/1.75 = 0.229$$

$$\sigma_Y = 0.3/1.75 = 0.171$$

The relative angular acceleration between the chaser and target is given by:

$$\dot{\omega}_{T_Z/C_Z} = \dot{\omega}_{T_Z} - \dot{\omega}_{C_Z}$$

Remembering that the target is not allowed to make any sudden movement changes during the docking sequence, $\dot{\omega}_{T_Z} = 0$. Therefore the disturbance angular acceleration is the angular acceleration of the chaser. The maximum torque around the chaser's z-axis is software limited to 0.086 Nm and the chaser's moment of inertia around its z-axis is estimated as 0.089 kg.m². So the standard deviation of the disturbance angular acceleration is chosen as:

$$\sigma_{\omega} = (0.086)/0.089 = 0.966$$

Therefore the process noise covariance matrix is:

$$\mathbf{Q}_{DF} = E\{(\mathbf{G}_1 \mathbf{w}_1)(\mathbf{G}_1 \mathbf{w}_1)^T\}$$

Resulting in a diagonal matrix whose diagonal is given by:

$$\text{diag}(\mathbf{Q}_{DF}) = \begin{bmatrix} 0 & 0 & \sigma_Z^2 & \sigma_Y^2 & 0 & \sigma_{\omega}^2 \end{bmatrix}$$

The above matrix is the continuous covariance matrix and has to be converted to its discrete equivalent, \mathbf{Q}_{DF_k} . If it is assumed that the process noise is white compared to the outer loop's sample period, T_{s_o} , then the approximation given by Franklin *et al.* [7] can be used. According to this, the diagonal of \mathbf{Q}_{DF_k} is given by:

$$\text{diag}(\mathbf{Q}_{DF_k}) = \begin{bmatrix} 0 & 0 & \sigma_Z^2 & \sigma_Y^2 & 0 & \sigma_{\omega}^2 \end{bmatrix} \left(\frac{2\tau_c}{T_{s_o}} \right)$$

Where τ_c is the time correlation of the noise and can be either measured or estimated. From section 6.4, $T_{s_o} = 0.1$ s and τ_c is estimated as 0.01 s.

9.5.2 Measurement noise

From the camera calibration results in section 8.6.1, the measurement noise covariance matrix is given by (in pixels squared):

$$\mathbf{R}_{DF_k} = \begin{bmatrix} (1.3288)^2 & 0 \\ 0 & (1.3288)^2 \end{bmatrix}$$

9.6 Conclusion

In this chapter Malan's filter was customised for the purposes of this thesis. The simulated and practical results will be shown and discussed in chapters 13 and 14, respectively. Note that in the rest of this thesis there will be referred to this modified version of Malan's EKF as "Malan's EKF".

Chapter 10

Inner Loop Controller

The purpose of the inner loop controllers is to maintain the accelerations and angular rate commanded by the outer loop controllers in the presence of external disturbances such as friction. Stated differently, the inner loop controllers must ensure that the platform looks like an ideal platform to the outer loop controllers. This chapter presents the design and results of the inner loop controllers.

10.1 Simulation and Controller Design Methodology

The principle used is to make the hovercraft's model as complex (or rather, realistic) as possible for simulation purposes, but to use a simplified model to design the controllers. Then, the controllers' performance is evaluated using the complex model of the hovercraft to see if the performance is satisfactory. If this is not the case, the assumptions of the simplified model must be re-evaluated for correctness and, if necessary, the complexity of the simplified model must be increased and the design process repeated.

Therefore, the simulation model includes:

- The effects of the sensor board's misalignment.
- The non-ideal sensor gains.
- The influence of the anti-aliasing filters.
- The effect of rotations on the accelerometers that are not mounted on the hovercraft's Centre of Gravity (CG).

For the controller the model is simplified:

- It ignores the effect of the sensor boards misalignment as the alignment is close to orthogonal.
- The non-ideal accelerometer gains are ignored as the calibration constants, K_x and K_y , are close to unity while the accelerometer outputs are very noisy due to the mechanical vibrations. On the other hand, the gyro's measurement will be corrected as its calibration constant, K_θ , is 0.9464 (5.36% error) while the sensor has very good noise characteristics.

- If the estimator's and controller's closed loop poles are placed at frequencies at least 5 times slower than the anti-aliasing filters' poles, then the influence of the anti-aliasing filters can be ignored.
- Due to the system's low bandwidth, the effect of angular accelerations and angular velocities on the accelerometers' measurements is negligible. (Especially if one considers how noisy the accelerometers' outputs are.)

The simulation's model consists of an ideal core from which ideal measurements can be obtained. These measurements are then contaminated with noise and the before mentioned effects to accurately simulate the practical measurements, before they are fed back to the controllers. Refer to appendix F for the complete hovercraft model used in the simulation.

10.2 The Model Used to Design the Controller

10.2.1 Another coordinate system

As defined in section 2.1, the chaser's body axis is fixed to the chaser's CG. Therefore, as the chaser's configuration changes, its CG and thus the chaser's body axis moves around. To make provision for this, a Body Fixed Reference Axis (x_B, y_B) is defined as shown in figure 10.1. (The reference axes run parallel to the chaser's body axes and the reference x-axis runs down the centre of the hovercraft.) By doing this, only the position of the CG in the reference axis has to be updated instead of redetermining the positions of all the actuators and all the sensors in the chaser's body axis every time the CG moves. Note that the reference has no z-axis as it is irrelevant in this 2D application and therefore any z-component will be assumed to be 0.

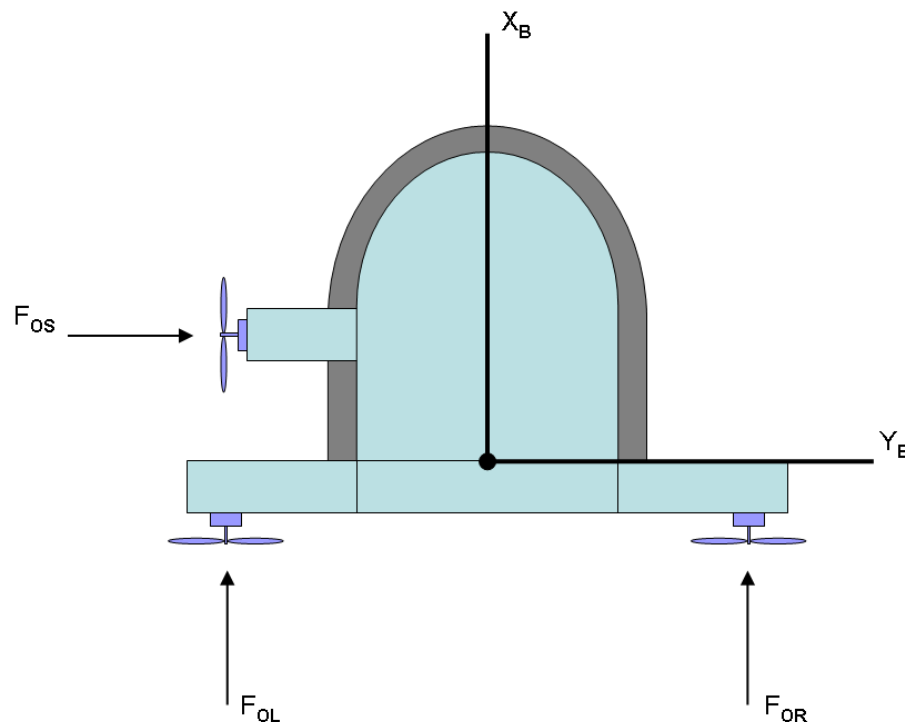


Figure 10.1: Chaser's Body Fixed Reference Axis.

10.2.2 The basic model

The hovercraft model used to design the controller is shown in figure 10.2. (Compare it to figure 10.1.)

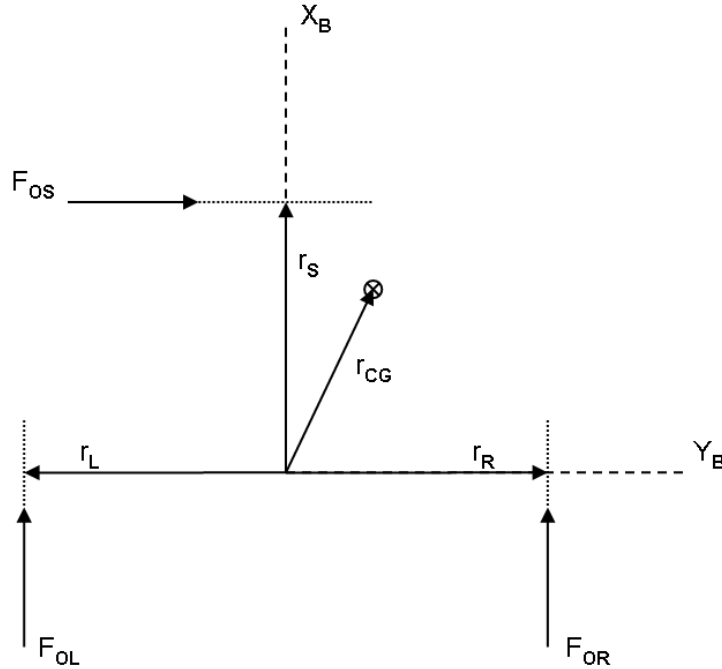


Figure 10.2: Basic Hovercraft Model.

Where:

- $\mathbf{F}_{OL} = [F_{OL}, 0, 0]^T$ is the output force of the fan on the left hand side of the hovercraft.
- $\mathbf{F}_{OR} = [F_{OR}, 0, 0]^T$ is the output force of the fan on the right hand side of the hovercraft.
- $\mathbf{F}_{OS} = [0, F_{OS}, 0]^T$ is the output force of the sideways fan on the hovercraft.
- $\mathbf{r}_L = [0, r_L, 0]^T$ is the position of \mathbf{F}_{OL} in the chaser's reference axis.
- $\mathbf{r}_R = [0, r_R, 0]^T$ is the position of \mathbf{F}_{OR} in the chaser's reference axis.
- $\mathbf{r}_S = [r_S, 0, 0]^T$ is the position of \mathbf{F}_{OS} in the chaser's reference axis.
- $\mathbf{r}_{CG} = [r_{CGx}, r_{CGy}, 0]^T$ is the position of the CG in the chaser's reference axis.

Note that it is assumed that the actuator output forces are parallel to the chaser's axes. For now, as the hovercraft moves relatively frictionless, the effect of friction is not modeled. According to Newton's 3rd law, the lift fan's prop will exert a torque on the chaser's body equal to the torque exerted on it, but in the opposite direction. It is assumed that this torque is small so that the resulting angular acceleration of the hovercraft is negligible. Although it is also assumed that the controllers' integrators will compensate for these effects, both effects could be modeled if it is found necessary.

The hovercraft model is described by the following continuous time differential equations:

$$\begin{aligned}\dot{F}_{OL} &= -\left(\frac{1}{\tau_L}\right)F_{OL} + \left(\frac{1}{\tau_L}\right)F_{IL} \\ \dot{F}_{OR} &= -\left(\frac{1}{\tau_R}\right)F_{OR} + \left(\frac{1}{\tau_R}\right)F_{IR} \\ \dot{F}_{OS} &= -\left(\frac{1}{\tau_S}\right)F_{OS} + \left(\frac{1}{\tau_S}\right)F_{IS} \\ \ddot{x}_{CG} &= \left(\frac{F_{OL} + F_{OR}}{m}\right) \\ \ddot{y}_{CG} &= \left(\frac{F_{OS}}{m}\right) \\ \ddot{\theta} &= -\left(\frac{r_L - r_{CGy}}{I_{ZZ}}\right)F_{OL} - \left(\frac{r_R - r_{CGy}}{I_{ZZ}}\right)F_{OR} + \left(\frac{r_S - r_{CGx}}{I_{ZZ}}\right)F_{OS}\end{aligned}$$

Where:

- F_{IL} , F_{IR} and F_{IS} are the input (commanded) forces to the system.
- F_{OL} , F_{OR} and F_{OS} are the output forces of the fans.
- τ_L , τ_R and τ_S are the time constants of the left, right and sideways fans, respectively. The left and right fans are of fan type 1, so that $\tau_L = \tau_R = 81.5$ ms. The sideways fan is of fan type 2 so that $\tau_S = 105$ ms.
- I_{ZZ} is the hovercraft's moment of inertia around the z-axis, and
- m is the hovercraft's mass.
- The acceleration of the hovercraft's CG along its x- and y-axis are \ddot{x}_{CG} and \ddot{y}_{CG} , respectively.
- $\ddot{\theta}$ is the angular acceleration of the hovercraft around its z-axis.

This model's continuous state space representation is given by:

$$\begin{bmatrix} \dot{F}_{OL} \\ \dot{F}_{OR} \\ \dot{F}_{OS} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -\left(\frac{1}{\tau_L}\right) & 0 & 0 & 0 \\ 0 & -\left(\frac{1}{\tau_R}\right) & 0 & 0 \\ 0 & 0 & -\left(\frac{1}{\tau_S}\right) & 0 \\ \frac{r_{CGy}-r_L}{I_{ZZ}} & \frac{r_{CGy}-r_R}{I_{ZZ}} & \frac{r_{CGx}-r_S}{I_{ZZ}} & 0 \end{bmatrix} \begin{bmatrix} F_{OL} \\ F_{OR} \\ F_{OS} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} \left(\frac{1}{\tau_L}\right) & 0 & 0 \\ 0 & \left(\frac{1}{\tau_R}\right) & 0 \\ 0 & 0 & \left(\frac{1}{\tau_S}\right) \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} F_{IL} \\ F_{IR} \\ F_{IS} \end{bmatrix}$$

With the outputs:

$$\begin{bmatrix} \ddot{x}_{CG} \\ \ddot{y}_{CG} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \left(\frac{1}{m}\right) & \left(\frac{1}{m}\right) & 0 & 0 \\ 0 & 0 & \left(\frac{1}{m}\right) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{OL} \\ F_{OR} \\ F_{OS} \\ \dot{\theta} \end{bmatrix}$$

Due to the form of the matrices, the 3 axes cannot be decoupled.

10.2.3 Using virtual forces and torques

The idea behind virtual forces and virtual torques is to think in terms of the resulting forces and torques produced by all the actuators, rather than in terms of the individual forces produced by each actuator.

After the control signals have been determined in terms of virtual forces and torques, the control signals are converted back to the individual actuator forces required.

In order to decouple the model from the previous section, define and use the following virtual forces and virtual torques:

$$\begin{bmatrix} \hat{F}_X \\ \hat{F}_Y \\ \hat{T} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ -(r_L - r_{CGy}) & -(r_R - r_{CGy}) & (r_S - r_{CGx}) \end{bmatrix} \begin{bmatrix} F_L \\ F_R \\ F_S \end{bmatrix}$$

Where \hat{F}_X and \hat{F}_Y is the virtual forces along the chaser's x- and y-axis, respectively. \hat{T} is the virtual torque around the chaser's z-axis. Define the matrix, \mathbf{V} , that converts the actual forces to virtual forces :

$$\mathbf{V} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ -(r_L - r_{CGy}) & -(r_R - r_{CGy}) & (r_S - r_{CGx}) \end{bmatrix}$$

Therefore, the matrix that converts the virtual forces into actual forces is given by:

$$\mathbf{V}^{-1} = \begin{bmatrix} \frac{-(r_R - r_{CGy})}{(r_L - 2r_{CGy} - r_R)} & \frac{(r_S - r_{CGx})}{(r_L - 2r_{CGy} - r_R)} & \frac{-1}{(r_L - 2r_{CGy} - r_R)} \\ \frac{(r_L - r_{CGy})}{(r_L - 2r_{CGy} - r_R)} & \frac{-(r_S - r_{CGx})}{(r_L - 2r_{CGy} - r_R)} & \frac{1}{(r_L - 2r_{CGy} - r_R)} \\ 0 & 1 & 0 \end{bmatrix}$$

Using virtual forces and torques, the state space representation now becomes:

$$\begin{bmatrix} \dot{\hat{F}}_{OX} \\ \dot{\hat{F}}_{OY} \\ \dot{\hat{T}}_O \\ \dot{\hat{\theta}} \end{bmatrix} = \begin{bmatrix} -\left(\frac{1}{\tau_x}\right) & 0 & 0 & 0 \\ 0 & -\left(\frac{1}{\tau_y}\right) & 0 & 0 \\ 0 & 0 & -\left(\frac{1}{\tau_T}\right) & 0 \\ 0 & 0 & \left(\frac{1}{I_{ZZ}}\right) & 0 \end{bmatrix} \begin{bmatrix} \hat{F}_{OX} \\ \hat{F}_{OY} \\ \hat{T}_O \\ \hat{\theta} \end{bmatrix} + \begin{bmatrix} \left(\frac{1}{\tau_x}\right) & 0 & 0 \\ 0 & \left(\frac{1}{\tau_y}\right) & 0 \\ 0 & 0 & \left(\frac{1}{\tau_T}\right) \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{F}_{IX} \\ \hat{F}_{IY} \\ \hat{T}_I \end{bmatrix}$$

With the outputs:

$$\begin{bmatrix} \ddot{x}_{CG} \\ \ddot{y}_{CG} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \left(\frac{1}{m}\right) & 0 & 0 & 0 \\ 0 & \left(\frac{1}{m}\right) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{F}_{OX} \\ \hat{F}_{OY} \\ \hat{T}_O \\ \hat{\theta} \end{bmatrix}$$

Where:

- \hat{F}_{IX} and \hat{F}_{IY} are the input (commanded) forces along the chaser's x- and y-axis , respectively.

- \hat{T}_I is the input (commanded) virtual torque along the chaser's z-axis.
- \hat{F}_{OX} , \hat{F}_{OY} and \hat{T}_O are the virtual forces and torque produced by the actuator output forces.
- τ_X , τ_Y and τ_T are the time constant of the virtual force along the x-axis, y-axis and the virtual torque around the z-axis, respectively. As the virtual force along the chasers x-axis is the sum of the left and right fans' output forces, it is assumed that the time constant $\tau_X = \tau_R = \tau_L = 81.5$ ms. Similarly, $\tau_Y = \tau_S = 105$ ms. The actuator configuration is as such that the torque is only produced by the left and right fans so that it is assumed that $\tau_T = \tau_R = \tau_L = 81.5$ ms.

This can now be decoupled into 3 linear systems:

x-axis:

$$\dot{\hat{F}}_{OX} = -\left(\frac{1}{\tau_X}\right)\hat{F}_{OX} + \left(\frac{1}{\tau_X}\right)\hat{F}_{IX}$$

$$\ddot{x}_{CG} = \left(\frac{1}{m}\right)\hat{F}_{OX}$$

y-axis:

$$\dot{\hat{F}}_{OY} = -\left(\frac{1}{\tau_Y}\right)\hat{F}_{OY} + \left(\frac{1}{\tau_Y}\right)\hat{F}_{IY}$$

$$\ddot{y}_{CG} = \left(\frac{1}{m}\right)\hat{F}_{OY}$$

z-axis:

$$\begin{bmatrix} \dot{\hat{T}}_O \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -\left(\frac{1}{\tau_T}\right) & 0 \\ \left(\frac{1}{I_{ZZ}}\right) & 0 \end{bmatrix} \begin{bmatrix} \hat{T}_O \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} \left(\frac{1}{\tau_T}\right) \\ 0 \end{bmatrix} \hat{T}_I$$

$$\dot{\theta} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{T}_O \\ \dot{\theta} \end{bmatrix}$$

The hovercraft's parameters were measured and are given in table 10.1.

Parameter	Value	Unit
I_{ZZ}	0.089	kg.m ²
m	1.75	kg
r_L	-0.23	m
r_R	0.23	m
r_S	0.14	m
r_{CG_x}	0.14	m
r_{CG_y}	0	m

Table 10.1: The hovercraft's parameters.

These values are substituted into the continuous differential equations and are then discretized using the inner loop sample period, T_{s_i} , of 20 ms. This results in:

x-axis:

$$\begin{aligned}\hat{F}_{OX}[k+1] &= (0.7824)\hat{F}_{OX}[k] + (0.2176)\hat{F}_{IX}[k] \\ \ddot{x}_{CG}[k] &= (0.5714)\hat{F}_{OX}[k]\end{aligned}$$

y-axis:

$$\begin{aligned}\hat{F}_{OY}[k+1] &= (0.8266)\hat{F}_{OY}[k] + (0.1734)\hat{F}_{IY}[k] \\ \ddot{y}_{CG}[k] &= (0.5714)\hat{F}_{OY}[k]\end{aligned}$$

z-axis:

$$\begin{aligned}\begin{bmatrix} \hat{T}_O[k+1] \\ \hat{\theta}[k+1] \end{bmatrix} &= \begin{bmatrix} 0.7824 & 0 \\ 0.1993 & 1 \end{bmatrix} \begin{bmatrix} \hat{T}_O[k] \\ \hat{\theta}[k] \end{bmatrix} + \begin{bmatrix} 0.2176 \\ 0.0254 \end{bmatrix} \hat{T}_I[k] \\ \hat{\theta}[k] &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{T}_O[k] \\ \hat{\theta}[k] \end{bmatrix}\end{aligned}$$

10.3 Controller and Estimator Design

10.3.1 Acceleration controllers

Initially, it was decided to design the estimators and controllers by following the guidelines that the estimators' closed loop poles must be at least 5 times slower than the anti-aliasing filter's poles, and that the controllers' closed loop poles must be at least 5 times slower than the estimators' slowest poles. As the anti-aliasing filter's cut off frequency is at $94.248 \text{ rad.s}^{-1}$ (15 Hz), the fastest possible closed loop controller poles are 3.77 rad.s^{-1} . Considering that for stability reasons the outer loop controllers' poles must be about 5 times lower than this, the resulting system will be very sluggish. The solution to this problem is to place faster inner loop poles by following a different approach.

As stated in the previous paragraph, when designing an estimator and controller, the design guideline is to place the estimator's close loop poles at a frequency about 5 times faster than the controller's closed loop poles so that the system's output would be dominated by the slower controller poles. However, when the measurement noise is very large, as is the case with the accelerometers, the position of the estimator and controller poles can be reversed so that the system's output is dominated by the slower estimator poles. [7]

As a starting point, a linear quadratic estimator was designed. From section 7.3.2, the standard deviation of measurement noise is 0.06 m.s^{-2} due to mechanical vibrations. The continuous process noise's standard deviation is overestimated as 0.2 N. Again, the continuous process noise variance is must be converted to its discrete equivalent. (See section 9.5.1) The resulting linear quadratic estimator closed loop poles are:

- $z = 0.7249$ for the x-axis estimator. ($\omega_n = 16.0861 \text{ rad.s}^{-1}$)

- $z = 0.7788$ for the y-axis estimator. ($\omega_n = 12.4971 \text{ rad.s}^{-1}$)

Even after the process noise was overestimated, the resulting estimator poles are relatively slow. This makes sense as the estimator will now rely more on the model than the very noisy measurement, resulting in the measurement being heavily filtered. Also, note that y-axis estimator is slower than the x-axis estimator. This is because the y-axis time constant of 105 ms is larger than the x-axis time constant of 81.5 ms so that the y-axis measurement will therefore contain less high frequencies and can therefore be filtered more than the x-axis measurement.

Both the x- and y-axis controllers have an integrator to ensure a zero tracking error¹ by compensating for the small disturbances due to friction. (For this to be entirely true, the sensor biases must be known.) It was decided, for each axis, to place the controller's 2 close loop poles at the same frequency as the estimator's pole in a sort of a butterworth configuration. It was found that a true butterworth configuration results in an output that is a bit under damped, so that the controllers poles were changed to have a damping ratio of 0.707 to compensate for this. The resulting closed loop controller poles are:

- $z = 0.7760 \pm 0.1797i$ for the x-axis controller.
- $z = 0.8249 \pm 0.1473i$ for the y-axis controller.

For these pole's the expected rise time is in the order of:

$$\begin{aligned} t_r &= \frac{1.8}{16.0891} \\ &= 0.112 \text{ s} \end{aligned}$$

for the x-axis controller, and:

$$\begin{aligned} t_r &= \frac{1.8}{12.4971} \\ &= 0.144 \text{ s} \end{aligned}$$

for the y-axis controller. Note that this equation only holds for dominant closed loop poles and is therefore only used to give an indication of what rise time to expect.

10.3.2 Angular rate controller

A linear quadratic estimator and controller was designed to control the angular rate around the hovercraft's z-axis. Using the gyro's noise standard deviation of $3.536 \times 10^{-3} \text{ rad.s}^{-1}$ (from section 7.3.2) and estimating the continuous process noise standard deviation to be 0.1 Nm, the resulting estimator poles are:

- $z = 0.2695 \pm 0.3313i$ ($\omega_n = 61.4836 \text{ rad.s}^{-1}$)

¹This is a type one control law that is suitable for tracking constant (step-like) setpoints with zero asymptotic error.

Although it would have been advised to place these poles at a frequency about 5 times slower than the anti-aliasing filter's poles at $94.248 \text{ rad.s}^{-1}$, these poles were tested in both simulation and practice, and works extremely well as the system is very responsive and the average angular rate follows the reference with an error of less than 10%. These estimator poles are much faster than the estimator poles for the x- and y-axis accelerations as the rate gyros output has very good noise characteristics.

The angular rate controller also has an integrator to ensure a zero tracking error by compensating for any small disturbance torques due to friction. (Again, this is only true if the sensor bias is known.) When the linear quadratic regulator (LQR) was designed, it was ensured that the closed loop system has an acceptable rise time while also ensuring a limited overshoot. For the cost function:

$$J = \sum_{k=0}^N \left(\mathbf{x}^T \mathbf{Q} \mathbf{x} + u^T R u \right)$$

The final weights:

$$\begin{aligned} \mathbf{Q} &= \mathbf{I}_{3 \times 3} \\ R &= 10 \end{aligned}$$

resulted in a rise time of about 0.165 s.

10.4 Simulation Results

A simulation of the system, controllers and estimators was done using MATLAB's Simulink. The model is shown in figure 10.3. All the controllers and estimators are implemented in a s-function block on the left hand side so that the code can easily be ported to the dsPIC30F5011. The 3 function blocks in the middle implements the actuator models from section 7.4. Lastly, the large block on the right hand side contains the hovercraft's kinematics, anti-aliasing filters and ADCs. This block also includes the effect of non-ideal sensor gains and the effect of rotations on the accelerometers that are not mounted on the hovercraft's CG. Here follow a summary of the simulation results:

- Figure 10.4 shows the response to a step input of 0.1 m.s^{-2} along the chaser's x-axis. The 10-90% rise time is about 0.13 s.
- Figure 10.5 shows the response to a step input of 0.1 m.s^{-2} along the chaser's y-axis. The 10-90% rise time is about 0.17 s.
- Figure 10.6 shows the response to a step input of 0.1 rad.s^{-1} . The 10-90% rise time is about 0.165 s.

All the simulation results are satisfactory with the rise times very close to their expected values. Even the acceleration controllers' outputs looks good if one take into account the amount of noise in the accelerometers outputs.

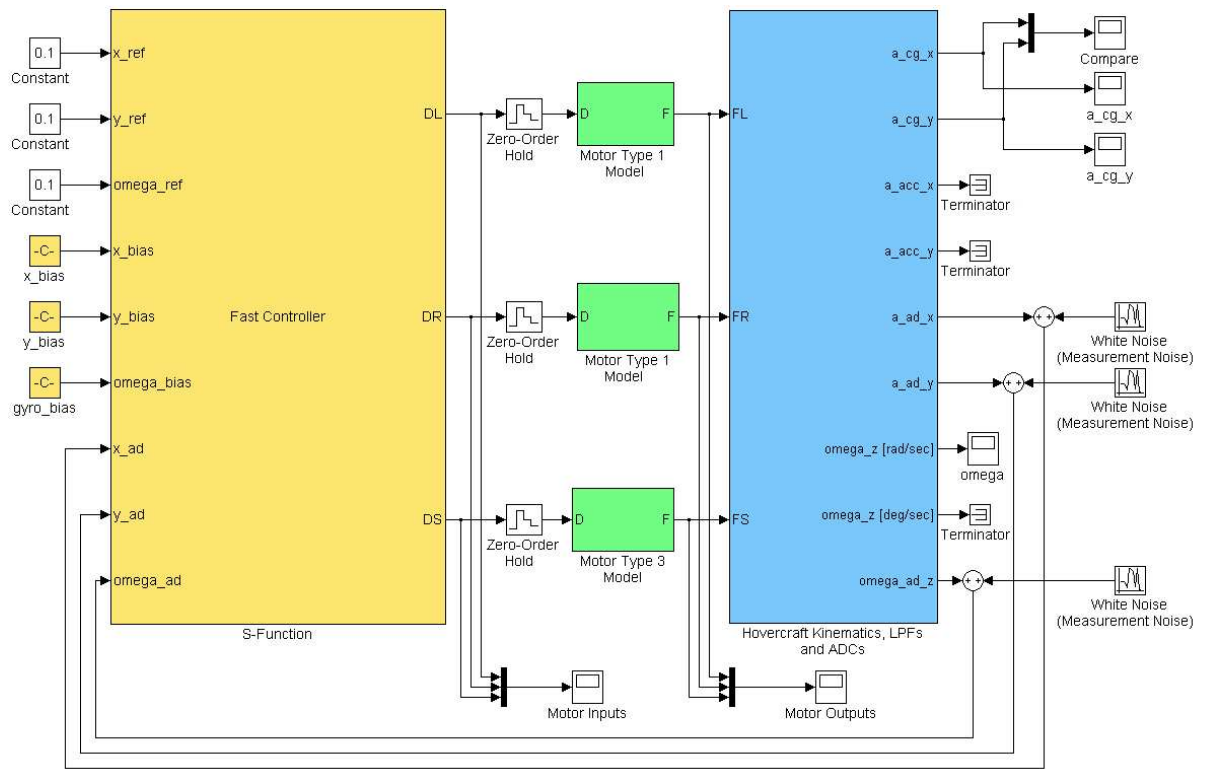


Figure 10.3: The inner loop simulation.

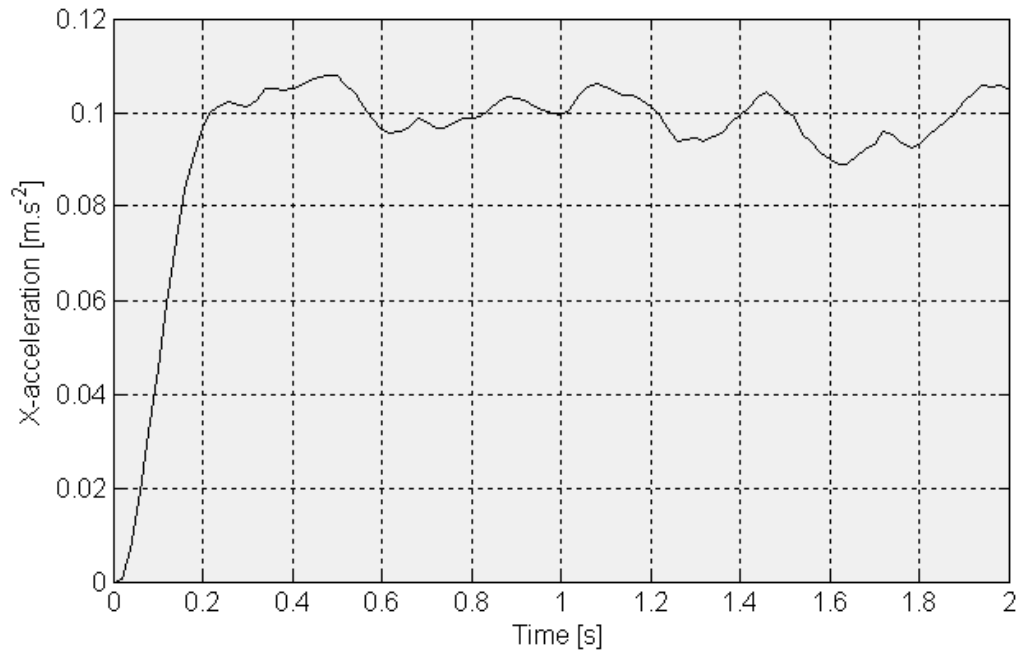


Figure 10.4: Acceleration of the CG along the x-axis.

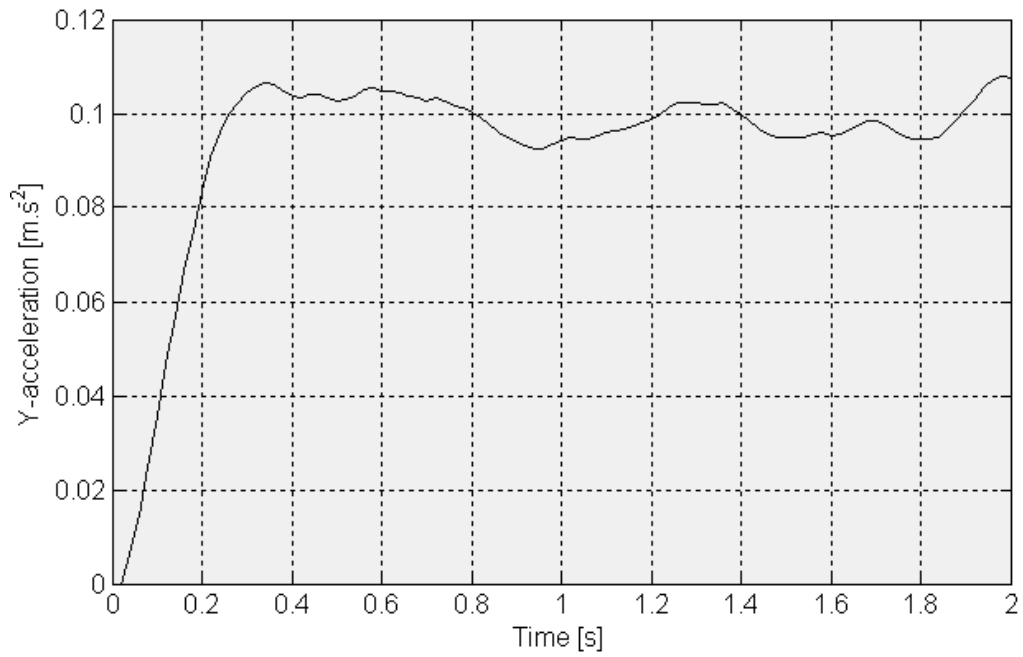


Figure 10.5: Acceleration of the CG along the y-axis.

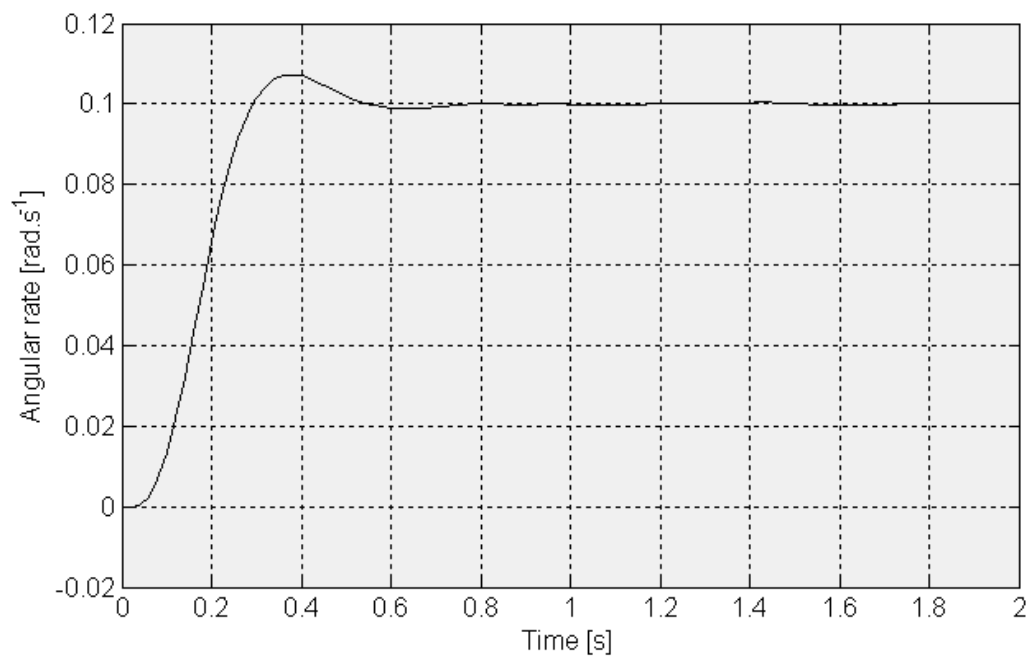


Figure 10.6: Angular rate around z-axis.

10.5 Practical Results

10.5.1 Acceleration controllers

A few practical experiments were done to test the acceleration controllers. Originally, the controllers' integrators were fed with the error between the measured accelerations and the reference accelerations. After this approach failed, the controllers were changed in that the integrators are now fed with the error between the estimated accelerations and the reference accelerations. Although this worked much better, it was still not satisfactory.

For brevity, only the response of the x- and y-acceleration controllers on a 0.1 m.s^{-2} step along the chaser's x-axis is shown in figures 10.7 and 10.9.

From figure 10.7 it can be seen that there is absolutely no correlation between the measured and estimated accelerations along the chaser's x-axis. Because of the large amount of noise in the accelerometer's measurement, the estimator only relies on the model. Therefore the estimated acceleration's behaviour is as expected.

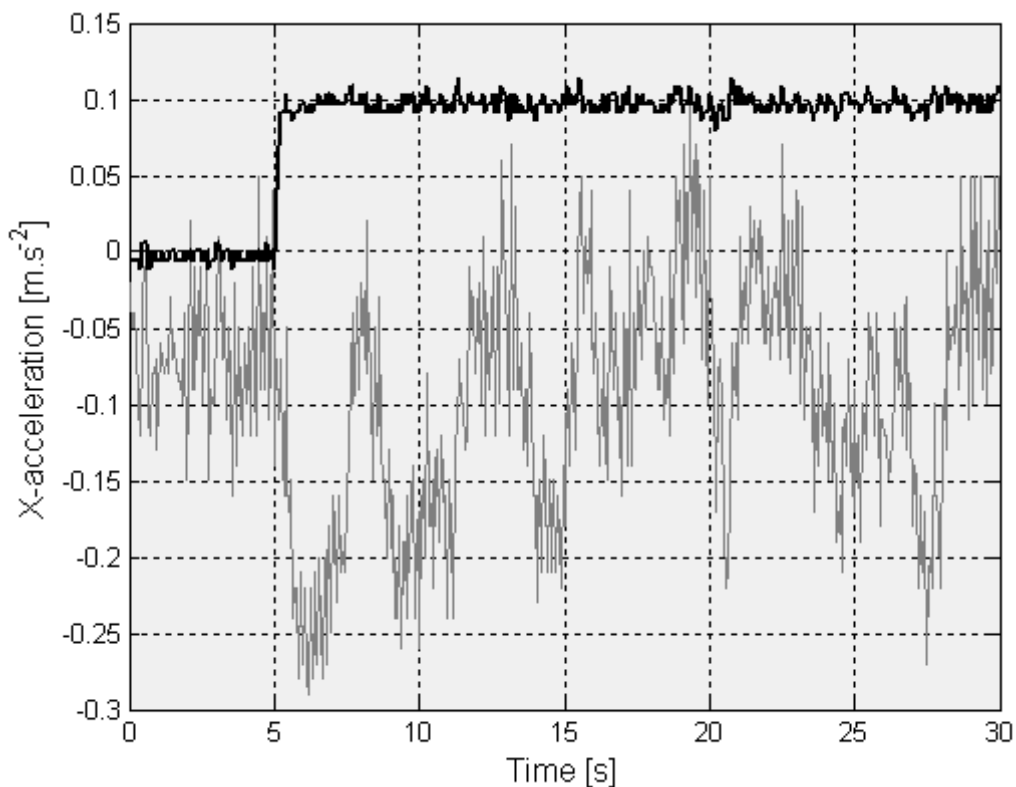


Figure 10.7: The estimated X-acceleration (black line) and the measured X-acceleration (gray line).

The deviations in the measured acceleration can be explained as follows:

1. The high frequency noise component is the result of the mechanical vibrations due to the fans' unbalanced props.

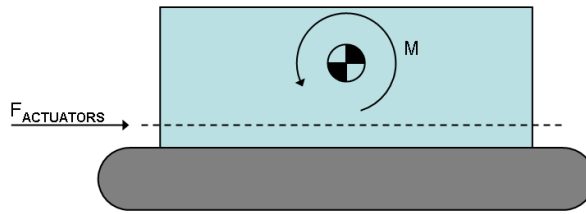


Figure 10.8: The tilting as result of the actuator forces.

2. As illustrated in figure 10.8, the force produced by the actuators causes a moment around the hovercraft's CG. This causes the hovercraft to tilt backwards. When this happens the gravitational acceleration couples into the accelerometer's measurement. From there the negative measurement seen in figure 10.7. (The slow oscillation is due to the air cushion's elasticity.)

Recall that at the beginning of this section it was said that feeding the controller's integrator with the error between the measured and reference acceleration did not work. It is this negative measurement obtained, when a positive acceleration is commanded, that causes the system to become unstable as it results in positive feedback.

By mounting the fans higher, the problem is reversed as the hovercraft now tilts forward. This causes the gravitational acceleration to couple into the accelerometers, in the opposite direction as before, resulting in a much larger measurement. The result of this is an oscillation: The fans switches on, the hovercraft tilts, an acceleration is measured that is much larger than the reference, the fans switches off and the cycle repeats.

Therefore it was decided to feed the controller's integrator with the error between the estimated acceleration and the reference acceleration.

Figure 10.9 shows the measured and estimated accelerations along the chaser's y-axis. From this figure the noise induced by the mechanical vibrations can clearly be seen as the measured acceleration should be zero. (Note that this is the measured acceleration after it as been filtered by both digital and analog filters.)

During each experiment, the hovercraft was accelerated from rest and the distance travelled and the time elapsed were recorded so that the average acceleration can be calculated.

For the 0.1 m.s^{-2} step along the chaser's x-axis, the hovercraft travelled 6.75 meters in 11.2 seconds. This implies an average acceleration of 0.108 m.s^{-2} that compares very well with the reference acceleration.

But for the 0.1 m.s^{-2} step along the chaser's y-axis the hovercraft only tilted while remaining stationary. (As the estimated acceleration is fed back instead off the measured acceleration, the controller is unaware of this.)

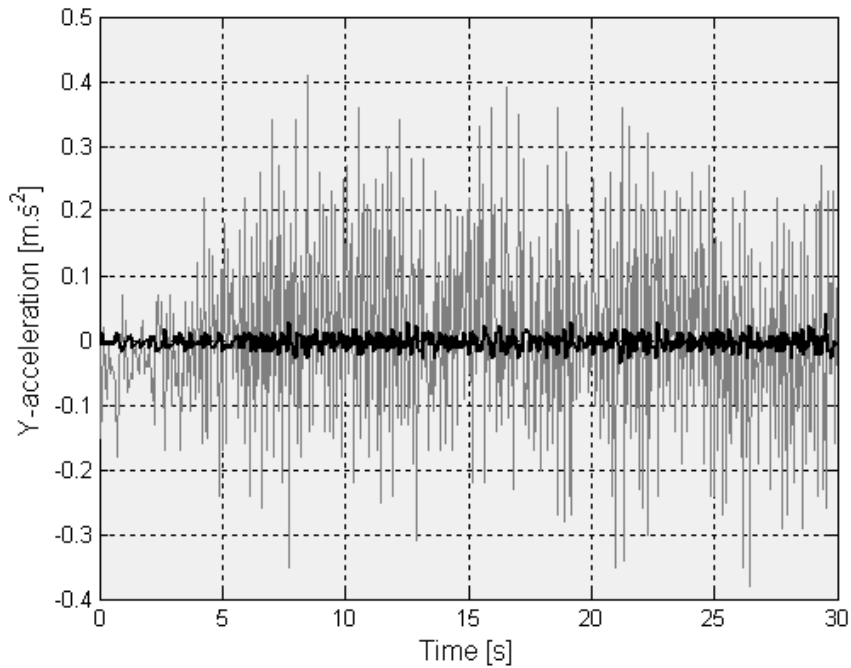


Figure 10.9: The estimated Y-acceleration (black line) and the measured Y-acceleration (gray line).

Attempts to improve the system includes limiting the actuator forces in software to reduce the tilting and trying to align the actuators that their thruster force do not produce any moments around the hovercraft's CG. Both attempts failed as very little tilting has an immense impact due to the gravitational acceleration's large magnitude. For example, the negative acceleration of -0.25 m.s^{-2} in figure 10.7 only requires a 1.46° tilt.

From this section it is clear that the accelerometers are not suitable for this application and should be replaced. This is addressed in section 10.6.

10.5.2 Angular rate controller

The angular rate estimator and controller works exceptionally well as can be seen in figure 10.10. As result of the fast estimator poles, the estimated angular rate closely follows the measured angular rate during the response on a 0.1 rad.s^{-1} step. During the experiment the hovercraft took 63.79 seconds to complete one full rotation. Therefore average angular rate was 0.098 rad.s^{-1} . This good result was obtained repeatedly as the angular rate sensor has good noise characteristics, is unaffected by the tilting and is less susceptible to the mechanical vibrations.

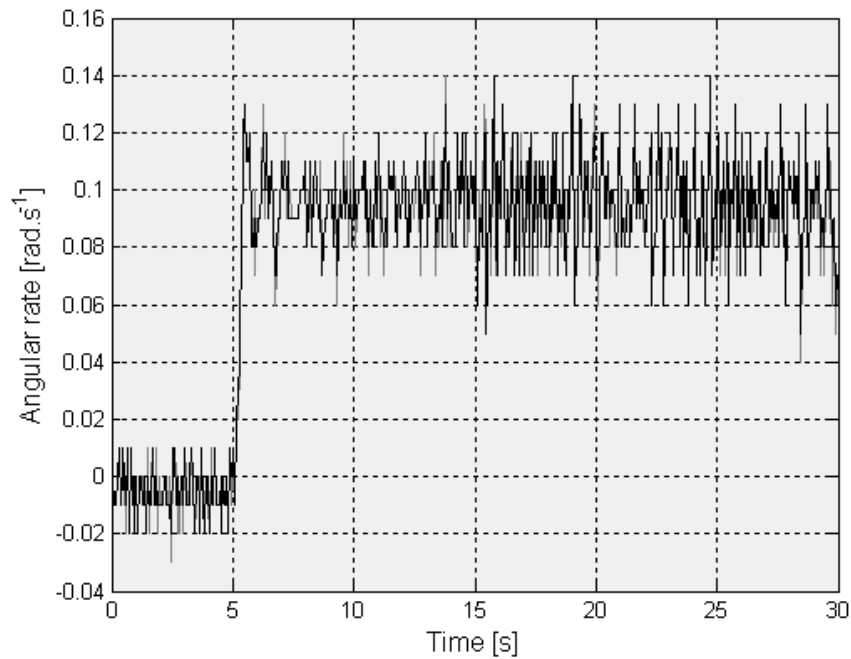


Figure 10.10: The estimated angular rate (black line) closely follows the measured angular rate (gray line).

10.6 Velocity Sensor and Controllers

In response to the problem stated in section 10.5.1, the accelerometers were replaced by a velocity sensor. The velocity sensor consists out of 2 potentiometers. Both potentiometers have no end stops so that their wipers can rotate freely. A wheel was mounted on the one potentiometer's axis and this wheel is dragged behind the hovercraft. The resulting sawtooth signal from this potentiometer's wiper is processed to determine the wheel's speed, $\|\mathbf{v}\|$. (See figure 10.11.) By using the second potentiometer to measure the wheel's angle behind the hovercraft, θ_V , the velocity, \mathbf{v} , can be divided into its components along the chaser's x- and y-axis.

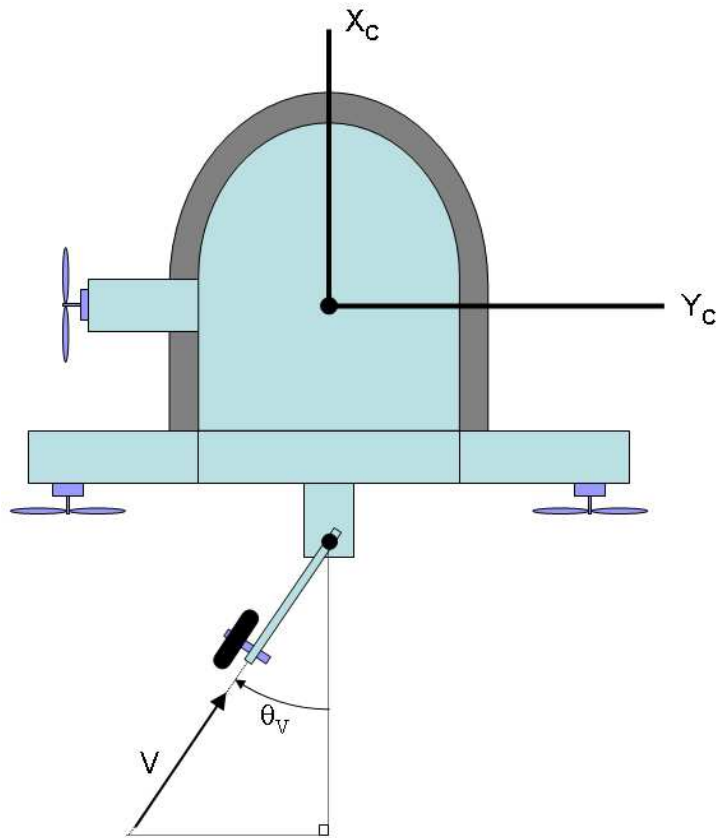


Figure 10.11: The velocity sensor.

10.6.1 New models for controller design

Still using the virtual forces introduced in section 10.2.3 the new continuous models used to design the controllers are:

x-axis:

$$\begin{bmatrix} \hat{F}_{OX} \\ \dot{v}_X \end{bmatrix} = \begin{bmatrix} -\left(\frac{1}{\tau_x}\right) & 0 \\ \left(\frac{1}{m}\right) & 0 \end{bmatrix} \begin{bmatrix} \hat{F}_{OX} \\ v_X \end{bmatrix} + \begin{bmatrix} \left(\frac{1}{\tau_x}\right) \\ 0 \end{bmatrix} \hat{F}_{IX}$$

$$v_X = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{F}_{OX} \\ v_X \end{bmatrix}$$

y-axis:

$$\begin{bmatrix} \hat{F}_{OY} \\ \dot{v}_Y \end{bmatrix} = \begin{bmatrix} -\left(\frac{1}{\tau_y}\right) & 0 \\ \left(\frac{1}{m}\right) & 0 \end{bmatrix} \begin{bmatrix} \hat{F}_{OY} \\ v_Y \end{bmatrix} + \begin{bmatrix} \left(\frac{1}{\tau_y}\right) \\ 0 \end{bmatrix} \hat{F}_{IY}$$

$$v_Y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{F}_{OY} \\ v_Y \end{bmatrix}$$

Using the values specified in section 10.2.3, these continuous models can be discretised using the inner loop sample period, T_{s_i} , of 20 ms:

x-axis:

$$\begin{bmatrix} \hat{F}_{OX}[k+1] \\ v_X[k+1] \end{bmatrix} = \begin{bmatrix} 0.7824 & 0 \\ 0.0101 & 1 \end{bmatrix} \begin{bmatrix} \hat{F}_{OX}[k] \\ v_X[k] \end{bmatrix} + \begin{bmatrix} 0.2176 \\ 0.0013 \end{bmatrix} \hat{F}_{IX}[k]$$

$$v_X[k] = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{F}_{OX}[k] \\ v_X[k] \end{bmatrix}$$

y-axis:

$$\begin{bmatrix} \hat{F}_{OY}[k+1] \\ v_Y[k+1] \end{bmatrix} = \begin{bmatrix} 0.8266 & 0 \\ 0.0104 & 1 \end{bmatrix} \begin{bmatrix} \hat{F}_{OY}[k] \\ v_Y[k] \end{bmatrix} + \begin{bmatrix} 0.1734 \\ 0.0010 \end{bmatrix} \hat{F}_{IY}[k]$$

$$v_Y[k] = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{F}_{OY}[k] \\ v_Y[k] \end{bmatrix}$$

10.6.2 Controller and estimator design

In section 9.5.1 it was seen that the maximum acceleration along the chaser's x-axis is slightly more than 0.2 m.s^{-2} and 0.15 m.s^{-2} along the chaser's y-axis.

By taking the above into account, a rise time can be calculated so that the actuators do not saturate during the maximum expected velocity step of 0.05 m.s^{-1} and 0.02 m.s^{-1} along the chaser's x- and y-axis, respectively. These rise times are:

$$t_{rx} = \frac{0.05}{0.2} = 0.250 \text{ s}$$

$$t_{ry} = \frac{0.02}{0.15} = 0.133 \text{ s}$$

For simplicity and the safety of the actuators, it was decided to use a rise time equal to twice the longest calculated rise time for both axes' controller design. This implies that that the closed loop system's dominant second order poles' natural frequency, ω_n , must be:

$$\omega_n = \frac{1.8}{2t_{rx}} = \frac{1.8}{2(0.25)} = 3.6 \text{ rad.s}^{-1}$$

The estimator's closed loop poles were placed at a frequency of 18 rad.s^{-1} (5 times lower than the anti-aliasing filter's frequency) with a damping ratio of 0.707. Each axis' controller has an integrator to ensure a zero tracking error. The 3 closed loop poles of the plant and controller is placed as follow:

- 2 Poles at the calculated frequency of 3.6 rad.s^{-1} with a damping ratio of 0.707.
- 1 Pole is placed on the real axis at -7.2 rad.s^{-1} so that it has a negligible influence on the closed loop system's output.

10.6.3 Simulation results

The MATLAB Simulink simulation shown in figure 10.3 was adapted to accommodate the velocity controllers. Figures 10.12 and 10.13 shows the simulation's results for a $0.05 \text{ m}\cdot\text{s}^{-1}$ step along the chaser's x- and y-axis, respectively. From these figures it can be seen that the rise time of the velocities along both axes is approximately 0.68 seconds. Although this is 36% longer than what was designed for, it is still adequate.

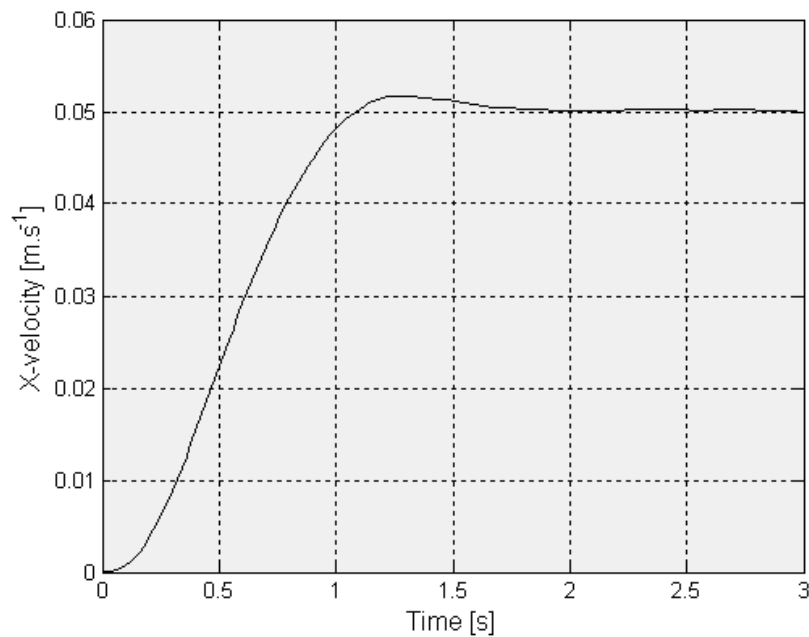


Figure 10.12: Velocity of the CG along the x-axis.

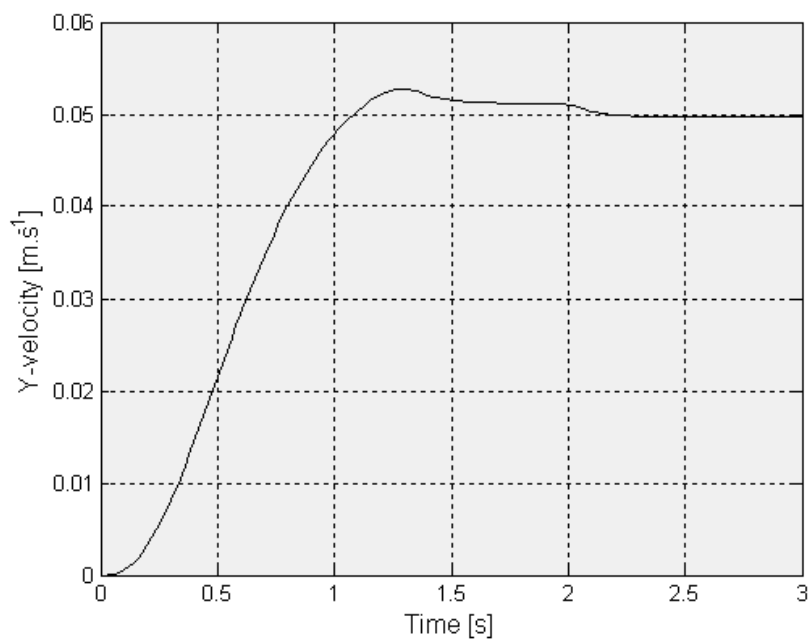


Figure 10.13: Velocity of the CG along the y-axis.

10.6.4 Practical results

Here follows the practical experiments results. Again, for brevity, only the response of a 0.05 m.s^{-1} step along the chaser's x-axis is shown.

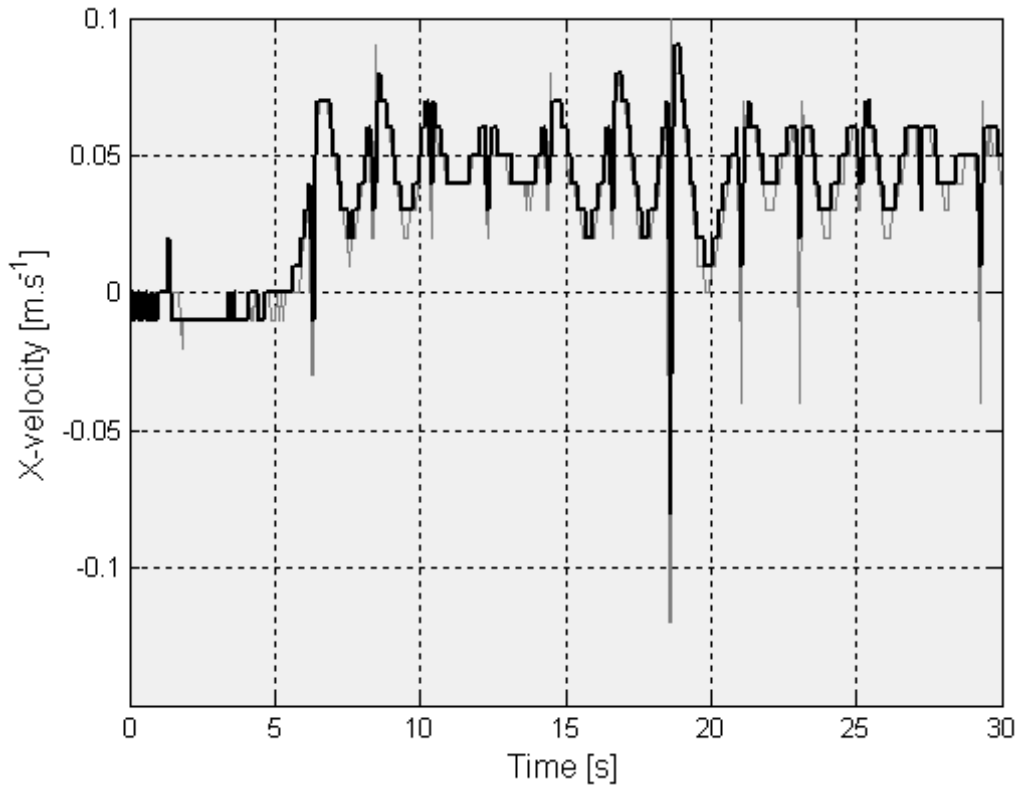


Figure 10.14: The estimated X-velocity (black line) and the measured X-velocity (gray line).

Figures 10.14 and 10.15 show the system's output. The clearly visible oscillation superimposed on the velocity measurement along the x-axis is due to the hovercraft's tilting. As the hovercraft tilts, the wheel is pushed to and fro, resulting in the oscillation seen. This confirms the explanation in section 10.5.1. Fortunately this sensor is not influenced by the mechanical vibration or gravitational acceleration.

During the experiments, the hovercraft traveled 2.1 meters in 46.08 seconds giving an average velocity of 0.04557 m.s^{-1} . This good result was obtained consistently for step commands along both the chaser's x- and y-axis.

From the figures it is estimated that the rise time is approximately 800 ms. This practical data will be used to model the inner loop controller and plant in the simulation discussed in chapter 13.

Note that the very coarse quantisation seen in figures 10.14 and 10.15 is only the result of the way data was sent from the inner loop controller to the computer during the tests. The inner loop controller makes use of doubles to store its values. To simplify the serial transmission of this data to the computer, each variable was multiplied by a 100, added to an offset and casted to an integer. On reception, this process was undone to extract the data. As the velocities are stored in m.s^{-1} , this process results in a resolution of 0.01 m.s^{-1} .

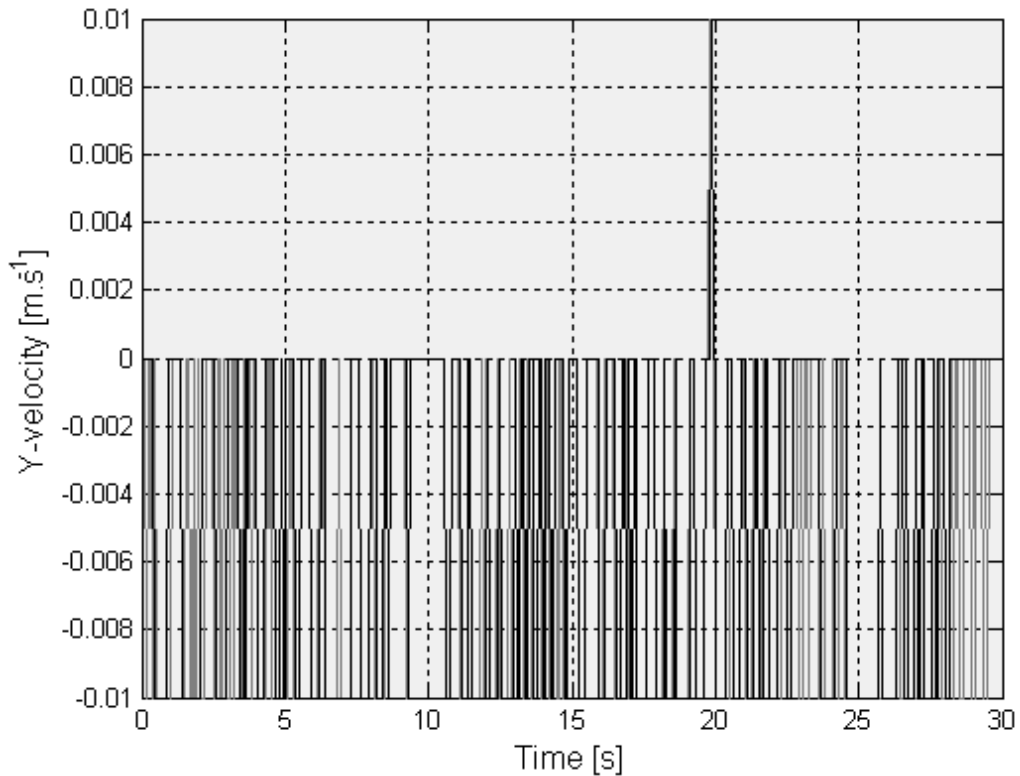


Figure 10.15: The estimated Y-velocity (black line) and the measured Y-velocity (gray line).

10.7 Conclusion

In this section the design of the different controllers were discussed. It was shown that the angular rate controller works very well, while it was necessary to replace the accelerometers and to rather make use of a velocity sensor. From the practical results it was seen that the velocity controllers works satisfactory.

The estimated velocities and angular rate are averaged over a 100 ms period and are then sent to the system EKF every 100 ms. This in turn provides the time base for the outer loop controllers and the EKFs.

Chapter 11

The System EKF

Several quantities need to be estimated in order to be able to execute the docking sequence. The System EKF is used for this purpose.

11.1 The State Vector

From section 2.1 it can be determined that the following state vector is required:

$$\bar{\mathbf{x}}_{sys} = \left[\bar{r} \quad \bar{\alpha} \quad \bar{\beta} \quad \bar{\omega}_{Tz} \right]^T$$

Where all the states are as defined in section 2.1 and $\bar{\omega}_{Tz}$ is the estimated angular velocity of the target around its own z-axis. Note that $\bar{\omega}_{Tz}$ is an unknown constant that has to be estimated.

11.2 Propagation

The system EKF is implemented as follows: The velocity sensor and rate gyro measurements¹ $[v_X \ v_Y \ \omega_{Cz}]^T$, received every 100 ms, are fed directly into the system EKF. As the velocity measurements, v_X and v_Y , and the rate gyro measurement, ω_{Cz} , are in the chaser's body axis, they have to be converted to the target's radial axis. This conversion is incorporated into the EKF. The non-linear continuous differential equations describing the propagation of the system states are given by:

$$\dot{\bar{\mathbf{x}}}_{sys} = \begin{bmatrix} -v_X \cos \bar{\alpha} + v_Y \sin \bar{\alpha} \\ \omega_{Cz} + \left(\frac{1}{\bar{r}}\right) (v_X \sin \bar{\alpha} + v_Y \cos \bar{\alpha}) \\ -\left(\frac{1}{\bar{r}}\right) (v_X \sin \bar{\alpha} + v_Y \cos \bar{\alpha}) - \bar{\omega}_{Tz} \\ 0 \end{bmatrix} + \mathbf{G}_2 \mathbf{w}_2 \quad (11.2.1)$$

¹Note that the estimated velocities and angular rate sent from the inner loop is seen as measurements by the system EKF.

Where:

$$\mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

And the process noise:

$$\mathbf{w}_2 = \begin{bmatrix} n_{\dot{r}} & n_{\dot{\alpha}} & n_{\dot{\beta}} \end{bmatrix}^T$$

Note that as the propagation is non-linear, it is integrated numerically.

11.3 Measurement Update

Directly after Malan's EKF has been updated, its state vector is processed to give the camera measurement, \mathbf{y}_{cm} .

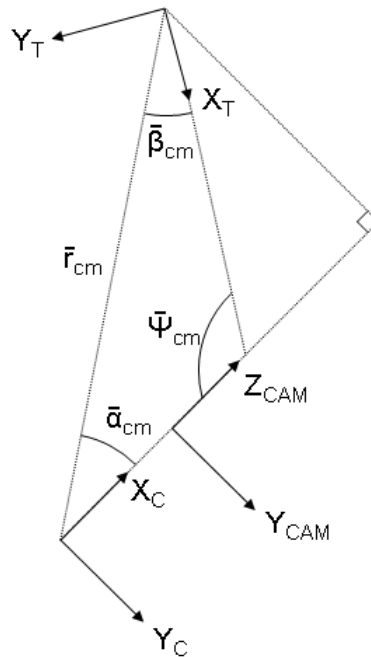


Figure 11.1: Processing the camera measurement.

Given that the position of the of the camera axis in the chaser's axis is given by:

$$\mathbf{r}_{CAM}^{CHR} = \begin{bmatrix} x_{CAM}^{CHR} & y_{CAM}^{CHR} & z_{CAM}^{CHR} \end{bmatrix}^T$$

And referring to figure 11.1, Malan's state vector is processed as follows:

$$\alpha_{cm} = \arctan \left(\frac{-(\bar{y} + y_{CAM}^{CHR})}{\bar{z} + x_{CAM}^{CHR}} \right)$$

$$\beta_{cm} = \pi - \alpha - \bar{\psi}$$

$$r_{cm} = \sqrt{(\bar{y} + y_{CAM}^{CHR})^2 + (\bar{z} + x_{CAM}^{CHR})^2}$$

$$\omega_{Tz(cm)} = \bar{\omega}_{Tz/Cz} + \overline{\omega_{Cz}}$$

Where $\overline{\omega_{Cz}}$ is the average angular rate of the chaser around its own z-axis between the current and previous photo. The camera measurement is then given by:

$$\mathbf{y}_{cm} = \left[\alpha_{cm} \quad \beta_{cm} \quad r_{cm} \quad \omega_{Tz(cm)} \right]^T$$

All the system EKF's states can be updated from this camera measurement so that the measurement matrix is a 4 by 4 identity matrix, $\mathbf{I}_{4 \times 4}$. If \mathbf{v}_2 is the measurement noise vector, the estimated camera measurement is given by:

$$\bar{\mathbf{y}}_{cm} = \mathbf{I}_{4 \times 4} \bar{\mathbf{x}}_{sys} + \mathbf{v}_2$$

This measurement update is done at a rate of about 2 Hz. As soon as a camera measurement becomes available, the next photo is taken. Due to the required processing, the delay between when a photo was taken and when the camera measurement becomes available is about 500 ms, but varies depending on how many centroids there are in the image. From this, the camera update rate is about 2 Hz.

11.4 Compensating for the Measurement Delay

Due to all the processing, there is a significant delay between when the photo is taken and when the camera measurement becomes available. This section describes how this delay is compensated for.

First some background on the implementation of Malan's EKF. Malan's EKF is fully isolated in the sense that it has no control inputs and is therefore oblivious to any control commands (accelerations and angular acceleration) that would otherwise have to be synchronised. Malan's EKF's implementation also assumes no measurement delay. This means that directly after Malan's EKF's state vector was updated, it reflects the state of the system at the time the photo was taken. As this updated state vector is used to give the camera measurement, the system EKF has to compensate for the total measurement delay, from when the photo was taken to when the camera measurement is received by the system EKF as is illustrated in figure 11.2.

There are 2 typical approaches to compensate for delays. The first is to insert delay states into the EKF system model and the second is to store (record) all the necessary information to compensate for the delay.

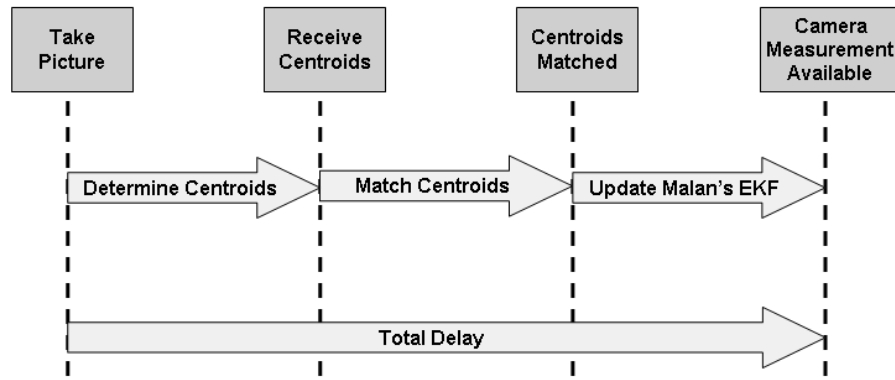


Figure 11.2: The measurement delay as seen by the system EKF.

For this application, the first method is unpractical as there will be too many states as:

$$\text{Number Of Delay States} = \frac{(4 \text{ States})(\text{Total Delay})}{(\text{Sample Period})}$$

The total delay is in the order of 500 ms and the sample period is 100 ms. Therefore one would require 20 delay states so that the system EKF's size is increased to 24 states. This EKF will be extremely computationally expensive especially as the inverse of a 24×24 matrix would have to be determined.

The other approach is to store the necessary data. When the photo is taken the system EKF's state vector is saved along with the state covariance matrix. Then the system EKF continues as normal while the inputs (velocity and gyro measurements) are recorded. When the camera measurement arrives, the system EKF first jumps back to the saved state, does the update and then quickly propagates the states using the recorded inputs. This method is much more practical and has the added advantage that it can compensate for variable measurement delays.

11.5 Noise Variances

11.5.1 Process noise

As the velocity and rate gyro measurements (or more accurately their estimated values) are the input to the system EKF, the process noise is related to their measurement noise. If one assumes that the angle, α , is accurately regulated to zero, then from equation 11.2.1:

$$\begin{aligned}\dot{\hat{r}} &= -v_X \\ \dot{\hat{\alpha}} &= \omega_{C_Z} + \left(\frac{v_Y}{\hat{r}}\right) \\ \dot{\hat{\beta}} &= -\omega_{T_Z} - \left(\frac{v_Y}{\hat{r}}\right)\end{aligned}$$

The standard deviation of the noise on the velocities and angular rates from the inner loop were calculated from practical data as 0.0199 m.s^{-1} and $0.0128 \text{ rad.s}^{-1}$, respectively. Taking the

before mentioned equations into account, the noise standard deviations were chosen as:

$$\begin{aligned}\sigma_{n_r} &= 0.0199 \\ \sigma_{n_k} &= 0.0328 \text{ using } r = 1 \text{ m} \\ \sigma_{n_{\dot{\beta}}} &= 0.0199 \text{ using } r = 1 \text{ m}\end{aligned}$$

Then the process noise is given by:

$$\mathbf{Q}_{sys} = E\{(\mathbf{G}_2 \mathbf{w}_2)(\mathbf{G}_2 \mathbf{w}_2)^T\}$$

Resulting in a diagonal matrix whose diagonal is given by:

$$\text{diag}(\mathbf{Q}_{sys}) = \begin{bmatrix} \sigma_{n_r}^2 & \sigma_{n_k}^2 & \sigma_{n_{\dot{\beta}}}^2 & 0 \end{bmatrix}$$

These are the sensors' continuous measurement noise and because it is integrated over one sample period, it has to be converted to its discrete equivalent. Using the same assumption as in section 9.5.1, the discrete noise covariance matrix is given by:

$$\text{diag}(\mathbf{Q}_{sys_k}) = \begin{bmatrix} \sigma_{n_r}^2 & \sigma_{n_k}^2 & \sigma_{n_{\dot{\beta}}}^2 & 0 \end{bmatrix} \left(\frac{2\tau_c}{T_{s_o}} \right)$$

11.5.2 Measurement noise

The measurement noise had to be determined using a simulation. From the simulation it was found that a good measurement covariance matrix is:

$$\text{diag}(\mathbf{R}_{sys_k}) = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix}$$

Note that these measurement noise variances do not reflect Malan's filter's state's noise variances. On average, the noise in Malan's filter is much less, but because of the current systems low bandwidth, the noise has been increased artificially so that the camera measurement is low pass filtered more to decrease the noise in the system.

11.6 Conclusion

This chapter described the system EKF that is used to estimate the necessary quantities required during the docking process. The simulated and practical results will be given in chapters 13 and 14, respectively.

Chapter 12

Outer Loop Controller And Guidance System

This chapter discusses the outer loop controllers and the guidance system.

12.1 Overview of the Outer Loop Controllers

There are 3 sets of controllers referred to as the “Far”, “Close” and “Final” controllers. Each set of controllers is scheduled by the guidance system according to the chaser’s position relative to the target satellite. Here is a brief description of each controller set:

- **Far Controllers** : These controllers slowly regulates the angle α to zero while following the given reference velocity. Note that the angle β is not regulated.
- **Close Controllers** : These controllers regulates the angle α more aggressively to zero while following the given reference velocity. The angle β is now also slowly regulated to zero.
- **Final Controllers** : The angles α and β are both aggressively regulated to zero while following the given reference velocity.

12.2 The Guidance System

This section gives a description of the guidance system used in the emulation. Note that $r_{FAR} = r_{RETREAT} = 1.5$ m and $r_{SAFE} = 1.0$ m. Referring to figure 12.1, here follows a brief description of the guidance system:

- **Far Approach** : When the chaser and target are far apart, $r > 1.5$ m, then the angle α will be slowly regulated to zero by the *far* controllers. At the same time a radial reference velocity of -0.02 m.s⁻¹ is given so that the chaser moves closer to the target. As the RGPS portion of this thesis was discarded, the camera is also initialised during this phase.
- **Close Approach** : When $1.0 < r < 1.5$, both angles α and β are regulated to zero using the *close* controllers. The chaser still approaches the target at the specified radial reference velocity of -0.02 m.s⁻¹.

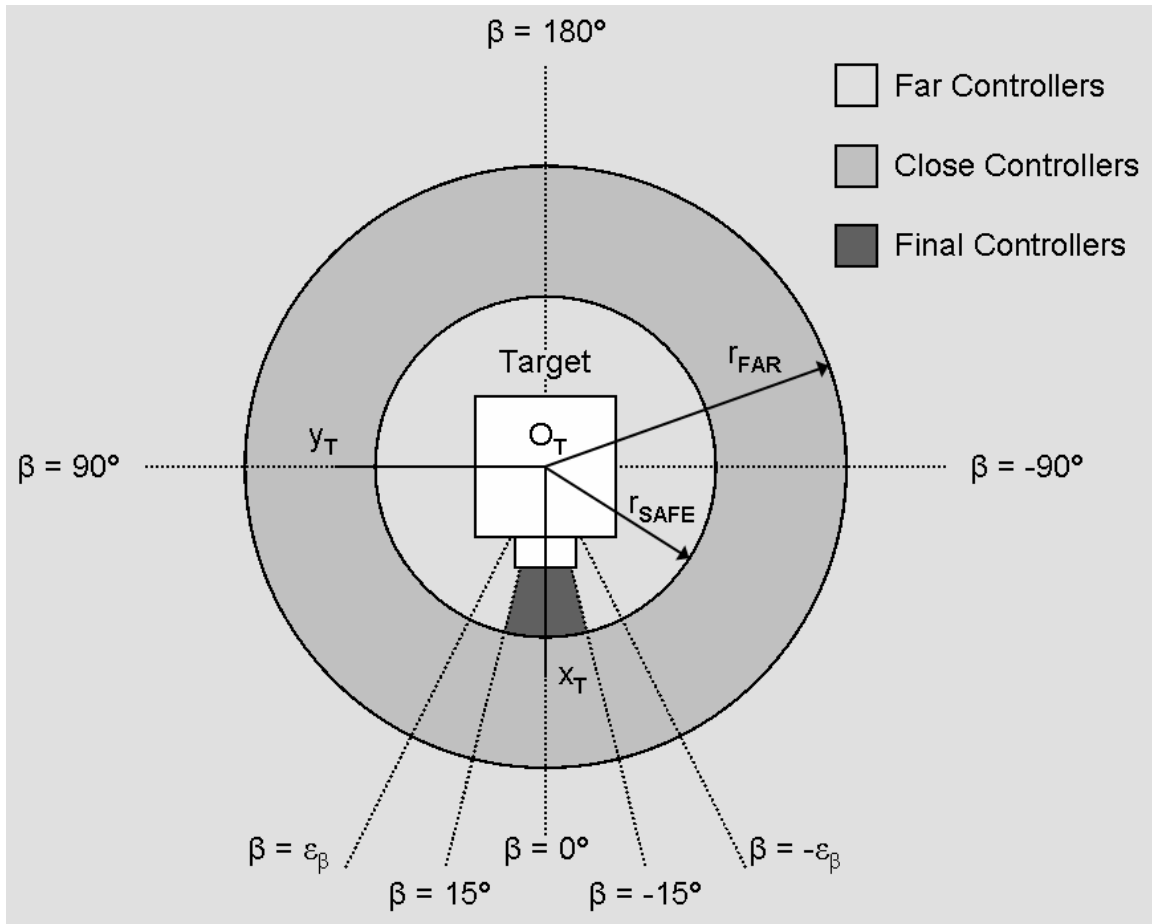


Figure 12.1: The regions where each controller set is used.

When $r = 1$ m, it has to be tested if it is safe to commence with the final approach. Theoretically, both angles α and β must be zero, but in practice one must specify an acceptable range for each angle. So while $|\alpha| > 15^\circ$ or $|\beta| > 15^\circ$, both angles will be regulated to zero by the *close* controllers. During this period the radial reference velocity is zero, so that the chaser remains on the border of the safety sphere.

- **Final Approach** : Once it is safe to dock, α and β will be very aggressively regulated to zero by the *final* controllers while following the radial reference velocity of $-0.02 \text{ m}\cdot\text{s}^{-1}$. When the distance between the chaser and target reaches 0.4 m, the docking sequence is complete. (This final distance was calculated.)

If any of the two angles' magnitude, $|\alpha|$ or $|\beta|$, exceeds 40° during the final approach, the docking sequence is aborted (i.e. $\epsilon_\alpha = \epsilon_\beta = 40^\circ$). The chaser will retreat to a distance of 1.5 m at a velocity of 0.03 m. This retreat is done using the *far* controllers as it was decided that the chaser should only increase the distance between itself and the target, while keep the camera pointing at the target. Once the retreat is completed, the docking sequence can be retried.

12.3 Models Used to Design the Outer Loop Controllers

The closed loop system, consisting of the inner loop controllers and the plant, is modeled as 3 decoupled 2nd order systems.

From practical data, such as in figure 10.14, it was determined that the rise time for a velocity step response along the chaser's x - and y -axis is approximately 0.8 s. Although one would be tempted to say that the system is under damped, the oscillations as seen in figure 10.14 are the result of the hovercraft's tilting. The oscillations are therefore noise and so the damping ratio has to be estimated. A damping ratio of 0.707 was chosen.

Similarly, for the angular rate, the rise time was obtained from practical data, such as in figure 10.10, as approximately 0.2 s. Due to the little overshoot, it is reasonable to assume that the damping ratio is also 0.707. The resulting models are shown in figure 12.2.

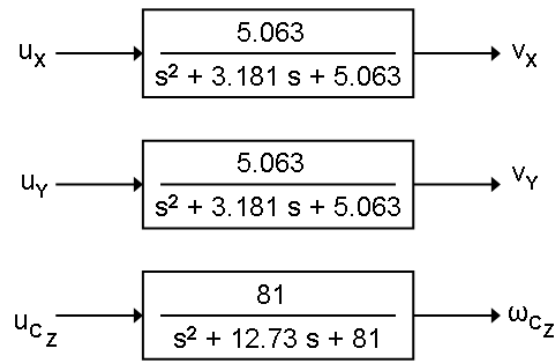


Figure 12.2: The 2nd order systems used to approximate the closed loop system consisting out of the inner loop controllers and plant.

Note that v_X , v_Y and ω_{C_Z} are the estimated velocities and angular rate that are sent from the inner loop's estimators to the system EKF. For clarity, the reference velocities and angular rate are u_X , u_Y and u_{C_Z} .

But the velocity models are in the chaser's Cartesian coordinate system, while the outer loop controllers are to be designed in terms of the target's radial coordinate system (r , β) and α . To solve this, a non-linear conversion block is used. The equations for these conversions can be determined from figure 12.3 and are given by equations 12.3.1 and 12.3.2.

$$\begin{bmatrix} v_X \\ v_Y \end{bmatrix} = \begin{bmatrix} -\cos \alpha & -r \sin \alpha \\ \sin \alpha & -r \cos \alpha \end{bmatrix} \begin{bmatrix} v_r \\ \omega_{C_T} \end{bmatrix} \quad (12.3.1)$$

$$\begin{bmatrix} v_r \\ \omega_{C_T} \end{bmatrix} = \begin{bmatrix} -\cos \alpha & \sin \alpha \\ -\left(\frac{1}{r}\right) \sin \alpha & -\left(\frac{1}{r}\right) \cos \alpha \end{bmatrix} \begin{bmatrix} v_X \\ v_Y \end{bmatrix} \quad (12.3.2)$$

These conversion blocks are then placed on either side of the 2nd order models from figure 12.2, so that the resulting encapsulated system are in terms of the target's radial coordinate system as shown in figure 12.4. Here u_r , u_{C_T} and u_{C_Z} are the reference radial velocity, reference angular rate of the chaser around the target and the reference angular rate of the chaser around

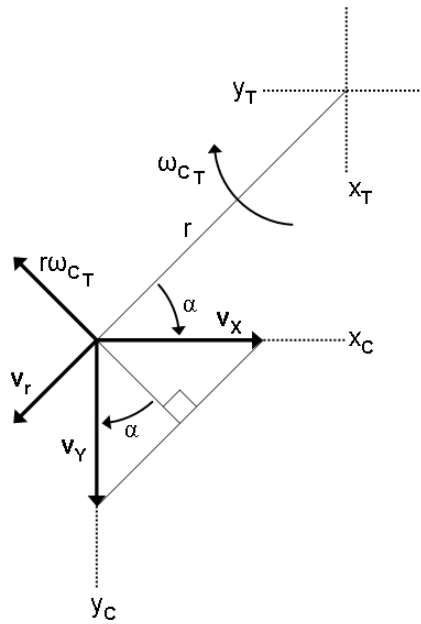


Figure 12.3: Diagram used to determine the conversion equations.

its own z-axis, respectively. Note that the relationship between the reference velocities u_r , u_{C_T} , u_x and u_y is also given by equations 12.3.1 and 12.3.2.

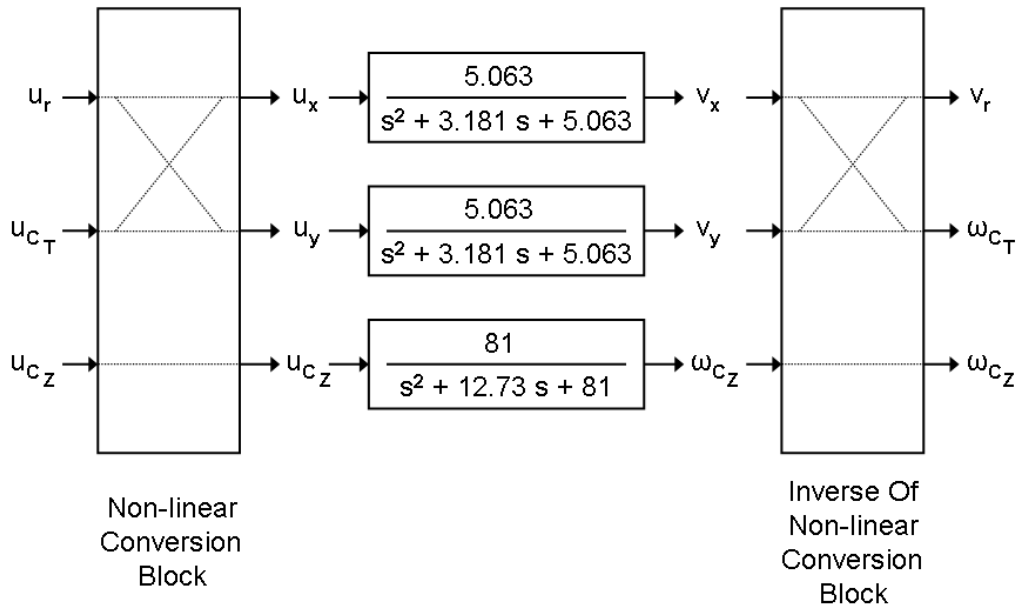


Figure 12.4: The 2nd order models encapsulated by the non-linear conversion blocks.

The “Non-linear conversion block” on the left of figure 12.4 implements equation 12.3.1. Similarly, the “Inverse of the non-linear conversion block” on the right implements equation 12.3.2. Now v_r , ω_{C_T} and ω_{C_z} are the measured radial velocity, measured angular rate of the chaser around the target and the measured angular rate of the chaser around its own z-axis, respectively. Note that this “Inverse of the non-linear conversion block” is already integrated into the system EKF. (See section 11.2.) Also note that both the reference, u_{C_z} , and the measurement,

ω_{C_Z} remains unaltered.

Now the system is described in terms of the target's radial axis, as was desired. However, due to the non-linear conversion blocks, the 3 previously decoupled models are now coupled again. Therefore in order to decouple them, for the controller design purposes, certain assumptions must be made.

If it is assumed that the controller that regulates α to zero works very well (i.e. $\alpha \approx 0$), then equations 12.3.1 and 12.3.2 reduces to:

$$\begin{bmatrix} v_X \\ v_Y \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -r \end{bmatrix} \begin{bmatrix} v_r \\ \omega_{C_T} \end{bmatrix} \tag{12.3.3}$$

$$\begin{bmatrix} v_r \\ \omega_{C_T} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -(\frac{1}{r}) \end{bmatrix} \begin{bmatrix} v_X \\ v_Y \end{bmatrix} \tag{12.3.4}$$

The block diagram from figure 12.4 then becomes:

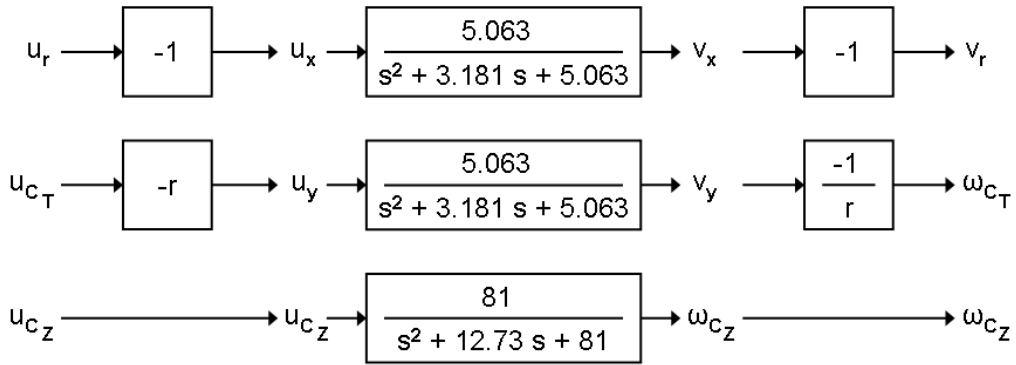


Figure 12.5: Under the assumption that $\alpha \approx 0$, the models are decoupled.

The gains on either side cancel out, resulting in the decoupled models used to design the outer loop controllers:

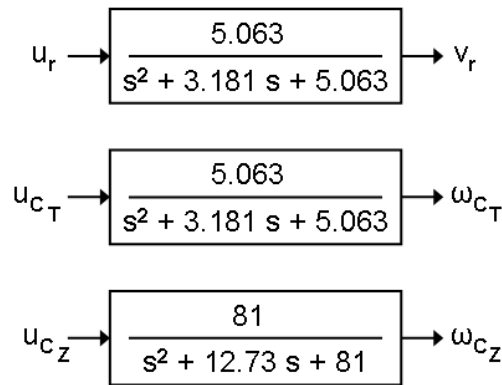


Figure 12.6: The decoupled models used to design the outer loop controllers.

Note that the poles of the first two systems in figure 12.6, i.e. the systems that correspond to v_r and ω_{C_T} , have a natural frequency, ω_n , of 2.25 rad.s^{-1} and that the poles of the system that corresponds to ω_{C_Z} have a natural frequency of 9 rad.s^{-1} .

12.4 The Outer Loop Controllers

12.4.1 Far controllers

12.4.1.1 The radial velocity v_r

For the radial velocity, the inner loop system is used as is:

$$u_r[k] = v_{r(ref)} = \begin{cases} -0.02 & \text{During far approach.} \\ 0.03 & \text{During retreat.} \end{cases}$$

12.4.1.2 The angle β

During the far approach, the angle β is not regulated so that:

$$u_{C_T}[k] = 0.0 \quad (12.4.1)$$

12.4.1.3 The angle α

The differential equation that describes the rate of change of the angle α is:

$$\dot{\alpha} = \omega_{C_Z} - \omega_{C_T} \quad (12.4.2)$$

From figure 12.6 and equation 12.4.2 the model used to design the controller for the angle α is:

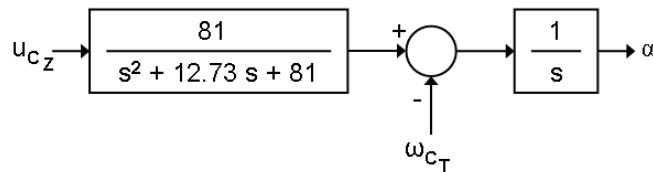


Figure 12.7: Model used to design the far controller for α .

If the dominant closed loop pole is placed at a frequency much lower than the rest, then the higher frequency poles can be ignored. Also during far approach $\omega_{C_T} = 0$ (from equation 12.4.1), so this controller can be designed using the model in figure 12.8.

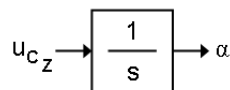


Figure 12.8: Simplified model used to design the far controller for α .

During the far approach, only a proportional controller will be used:

$$u_{Cz}[k] = -k_{\alpha}\bar{\alpha}[k]$$

Where $\bar{\alpha}[k]$ is the angle α as estimated by the system EKF and k_{α} is the proportional gain.

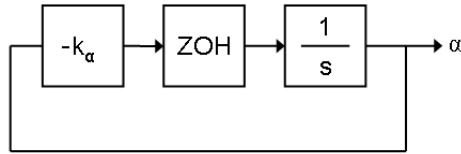


Figure 12.9: The far controller for the angle α .

It can be shown that the closed loop system in figure 12.9 is a first order low pass filter with a time constant of:

$$\tau_{\alpha} = \frac{-T_{s_0}}{\ln(1 - k_{\alpha}T_{s_0})}$$

Where τ_{α} is the low pass filter's time constant and T_{s_0} is the outer loop's sample period (100 ms). For such a first order low pass filter, it is generally assumed that the filter's output reaches its final value in 5 time constants. If it is chosen that the angle α must be regulated to zero in 20 seconds, then:

$$\begin{aligned} 20 &= 5\tau_{\alpha} \\ &= \frac{-5T_{s_0}}{\ln(1 - k_{\alpha}T_{s_0})} \end{aligned}$$

From this:

$$k_{\alpha} = 0.2469$$

Note that the resulting closed loop pole's frequency is 0.25 rad.s^{-1} . This frequency is 36 times lower than the frequency of the open loop poles of the 2nd order system in figure 12.7, so that these much faster poles can be ignored.

12.4.2 Close controllers

12.4.2.1 The radial velocity v_r

Again, for the radial velocity, the inner loop system is used as is:

$$\begin{aligned} u_r[k] &= v_{r(ref)} \\ &= -0.02 \end{aligned}$$

12.4.2.2 The angle β

To design the controller, first consider the differential equation that describes the rate of change of the angle β :

$$\dot{\beta} = \omega_{C_T} - \omega_{T_Z} \tag{12.4.3}$$

From figure 12.6 and equation 12.4.3 the model to design the controller for the angle β is:

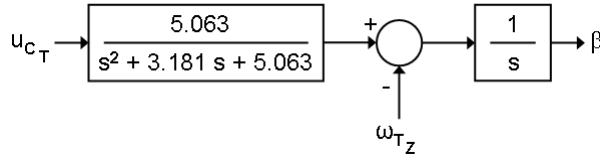


Figure 12.10: Model used to design the close controller for β .

The controller that will regulate the angle β to zero also contains an integrator to ensure a zero tracking error. In order to ignore the faster poles of the 2nd order system in figure 12.10, the dominant closed loop poles are placed at a frequency that is 10 times lower than the 2nd order system's poles. The simplified model used to design the controller for β is shown in figure 12.11.

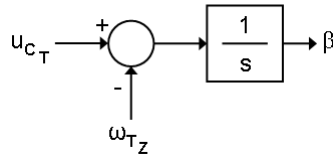


Figure 12.11: Simplified model used to design the close controller for β .

The simplified model is therefore described by:

$$\dot{\beta} = u_{C_T} - \omega_{T_Z} \tag{12.4.4}$$

From equation 12.4.4 it can be seen that there has to be compensated for ω_{T_Z} . This can easily be done as ω_{T_Z} is estimated by the system EKF.

Define:

$$u_{C_T}^* = u_{C_T} - \hat{\omega}_{T_Z} \tag{12.4.5}$$

The continuous state space model is then:

$$\dot{\beta} = u_{C_T}^*$$

So that the discrete state space model is:

$$\beta[k+1] = \beta[k] + T_{s_0} u_{C_T}^*[k]$$

As stated before, the dominant closed loop poles are placed at a frequency 10 times lower than the 2nd order system's poles. (From figure 12.10.) The dominant closed loop poles' frequency is therefore $\omega_n = 0.225 \text{ rad.s}^{-1}$. This implies a 10-90% rise time of 8 seconds. Chosen a damping ratio of 0.707 results in the following closed loop poles:

$$\begin{aligned} s_{CL_\beta} &= -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \\ &= -0.1591 \pm j0.1591 \end{aligned}$$

So that the discrete closed loop poles are:

$$\begin{aligned} z_{CL_\beta} &= e^{s_{CL_\beta} T_{s_0}} \\ &= 0.9841 \pm j0.0157 \end{aligned}$$

For the control law:

$$u_{C_T}^*[k] = -k_\beta \bar{\beta}[k] - k_{i_\beta} \beta_i[k] \quad (12.4.6)$$

Where k_β is the proportional gain, $\bar{\beta}[k]$ is the angle β as estimated by the system EKF, k_{i_β} is the controller's integrator's gain and $\beta_i[k]$ is the controller's integrator state. Using MATLAB's *place* function, the gains are calculated as:

$$\begin{aligned} k_\beta &= 0.3181 \\ k_{i_\beta} &= 0.0050 \end{aligned}$$

From equations 12.4.5 and 12.4.6 the control law is:

$$u_{C_T}[k] = -k_\beta \bar{\beta}[k] - k_{i_\beta} \beta_i[k] + \bar{\omega}_{T_Z}[k]$$

Where $\bar{\omega}_{T_Z}[k]$ is ω_{T_Z} as estimated by the system EKF.

Now there is only one more aspect of this controller that has to be addressed. As was shown before, the closed loop system's 10-90% rise time is 8 seconds. This is undesired as the angle β can be anything from -180° to 180° and it is impossible for the hovercraft to move halfway around the target in 8 seconds.

To solve this, the magnitude of the error (i.e. $\bar{\beta}[k]$) is clipped at 0.5° , so that the rate at which the hovercraft moves around the target is limited. This angle of 0.5° was determined using the simulation discussed in chapter 13. Using this simulation, the limit on the error was adjusted until the commanded actuator forces were within acceptable limits.

12.4.2.3 The angle α

The angle α is now regulated more aggressively to zero. To ensure a zero tracking error, the controller now also has a slow integrator. In order to ignore the faster 2nd order poles in figure 12.7, the dominant closed loop poles must be slower than 0.9 rad.s^{-1} . By doing this, the simplified model shown in figure 12.12 can be used to design the controller.

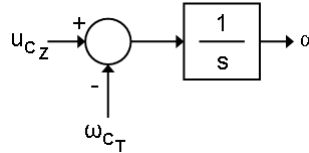


Figure 12.12: Simplified model used to design the close controller for α .

From figure 12.12 it can be seen that there has been compensated for ω_{CT} . At first it seems that this necessitates the estimation of ω_{CT} . This in turn requires that the system EKF has to be adapted. But closer inspection reveals that this is not necessary. The value of ω_{CT} will slowly increase from zero to a final value of ω_{TZ} as β is regulated to zero. Therefore ω_{CT} can be seen as a relatively constant offset and will automatically be compensated for by the controller's integrator.

The selected closed loop poles have a damping ratio of 0.707 to ensure a limited overshoot, and have a natural frequency of 0.36 rad.s^{-1} so that the 10-90% rise time is 5 seconds. Therefore the continuous closed loop poles are:

$$\begin{aligned} s_{CL\alpha} &= -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \\ &= -0.2545 \pm j0.2545 \end{aligned}$$

So that the discrete closed loop poles are:

$$\begin{aligned} z_{CL\alpha} &= e^{s_{CL\alpha}T_{s0}} \\ &= 0.9746 \pm j0.0248 \end{aligned}$$

The resulting control law is given as:

$$u_{Cz}[k] = -k_{\alpha}\bar{\alpha}[k] - k_{i_{\alpha}}\alpha_i[k]$$

Where $k_{i_{\alpha}}$ is the controller's integrator's gain and $\alpha_i[k]$ is the controller's integrator state. Using MATLAB's *place* function, the controller gains are:

$$\begin{aligned} k_{\alpha} &= 0.5089 \\ k_{i_{\alpha}} &= 0.0126 \end{aligned}$$

12.4.3 Final controllers

The final controllers are identical to the close controllers except that the limit on the error in the angle β is increased to 5° . This is to regulate the angle β more aggressively to zero. Although this might result in the use of large actuator forces, these large forces (if used) will only be used in short bursts.

12.5 Conclusion

In this chapter the design of the outer loop controllers were discussed. The controllers' simulated and practical results are shown in chapters 13 and 14, respectively.

Chapter 13

Simulation of the Docking Sequence

This chapter gives an overview of the docking sequence simulation that was written in C.

13.1 Overview

The purpose of this simulation was to implement, test and debug the code that will eventually run on the hovercraft's OBC. Because of this it was decided to use a layered software topology.

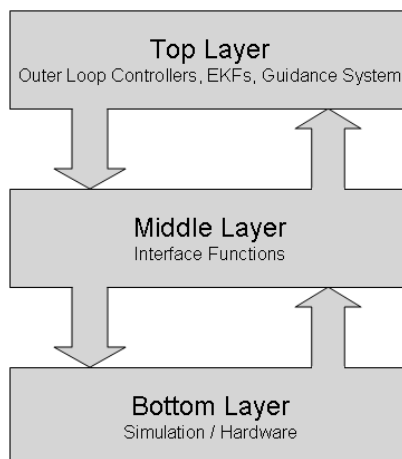


Figure 13.1: Graphical representation of the software layers.

From figure 13.1 it can be seen that the simulation software is divided into 3 layers:

- The **bottom layer** forms the base of the simulation and simulates the hardware.

To simulate the camera, the light model from chapter 4 is used to generate the ideal centroids before applying radial distortion to these centroids.

The closed loop system consisting of the plant and inner loop controllers are also simulated using the models from figure 12.2.

Different noise sources are also modeled. This includes the noise from the velocity sensor and gyros. (Noise with a Gaussian distribution is generated using the central limit theorem stating that the resulting distribution of the sum of several statistically independent

random variables approaches a Gaussian distribution. [23]) Pixel noise is implicitly simulated due to round off errors that occur when the floating point coordinates are cast to integer coordinates.

- The **top layer** consists of the outer loop controllers, the centroid identification and tracking unit, EKFs, light model and guidance system. (Everything inside the OBC block in figure 6.1.)
- The **middle layer** contains the interface functions. All the data and commands between the top and bottom layer are send via these interface functions.

Using this layered approach, the process of porting the software from the simulation to the hardware is simplified. Initially, for simulation purposes, the interface functions only update the necessary simulation variables. After the code has been debugged, the simulation (bottom) layer is replaced with the actual hovercraft. To accommodate this, only the interface functions has to be modified.

13.2 Simulation Parameters

This section gives the initial values of the simulation parameters.

- $r = 1.868$ m.
- $\alpha = -0.271$ rad = -15.52° .
- $\beta = 1.842$ rad = 105.52° .
- $\mathbf{r}_{CAM}^{CHR} = [0.13, 0.0, 0.0]^T$.
- The target's angular velocity, ω_{Tz} , was set to -0.007 rad.s $^{-1}$.
- Malan's EKF's initial state vector = $[3.0, 1.0, 0.0, 0.0, 0.0, 0.0]^T$.
- $diag(\mathbf{Q}_{DF_k}) = \begin{bmatrix} 0 & 0 & \frac{1}{5}(0.229)^2 & \frac{1}{5}(0.171)^2 & 0 & \frac{1}{5}(0.966)^2 \end{bmatrix}$
- $diag(\mathbf{R}_{DF_k}) = \begin{bmatrix} 1.3288^2 & 1.3288^2 \end{bmatrix}$
- The system EKF's initial state vector = $[3.0, 0.0, 0.0, 0.0]^T$.
- $diag(\mathbf{Q}_{sys_k}) = \begin{bmatrix} \frac{1}{5}(0.0199)^2 & \frac{1}{5}(0.0328)^2 & \frac{1}{5}(0.0199)^2 & 0 \end{bmatrix}$
- $diag(\mathbf{R}_{sys_k}) = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix}$

Note that although the safety sphere's radius is 1 m, hysteresis of 0.1 meters was inserted. If this is not done, then if r slightly overshoots (entering the safety sphere while it is not safe to dock) the guidance system will abort the docking sequence. As result the effective r_{SAFE} is 1.1 m as can be seen in figure 13.9.

13.3 Simulation Results

Figure 13.2 gives the chaser's trajectory in the simulation. The direction of the chaser's positive x-axis is given by the black arrows and the target's position is represented by the circle at (0, 0).

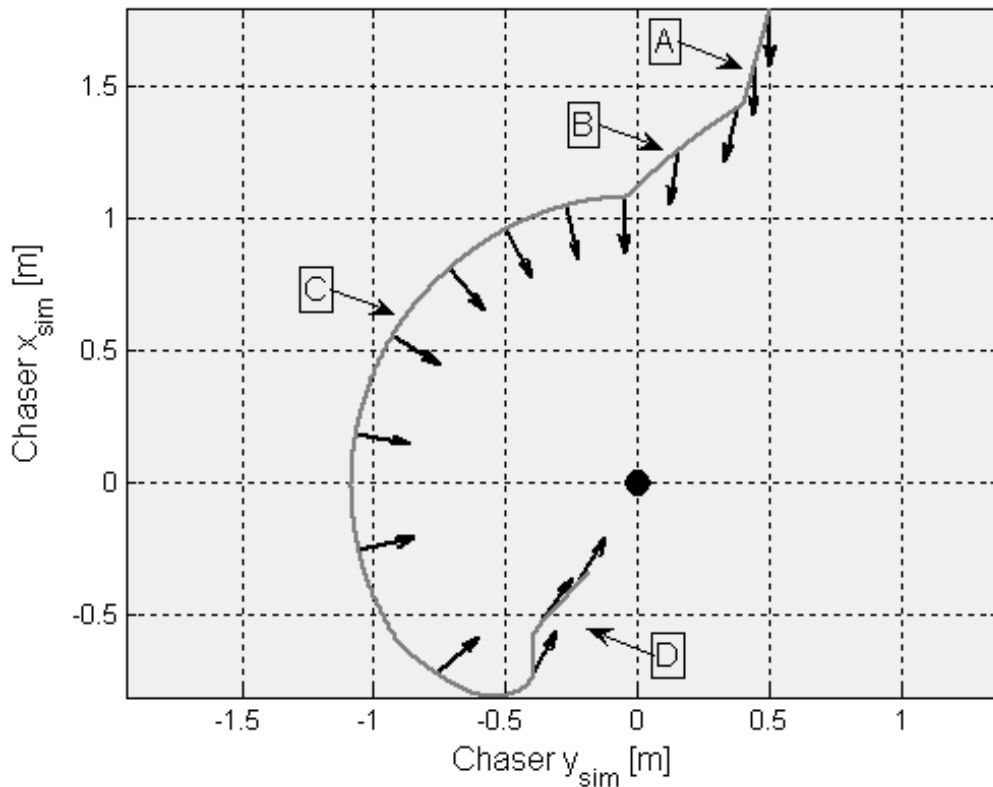


Figure 13.2: The chaser's trajectory (gray line) in the simulation. The direction of the chaser's positive x-axis is given by the black arrows.

Referring to figure 13.2:

- During line segment A, the far controllers point the docking mechanism and camera at the target by regulating α to zero. This is done while slowly decreasing the distance, r , between the chaser's and target's body axes' origins.
- As soon as the r reaches 1.5 m, the close controllers take over. These controllers keep α to zero while decreasing r . They also start to regulate β to zero. This gives rise to the line segment B.
- Then the distance r reaches 1.1 m (the effective radius of the safety sphere) before it is safe to dock. Line segment C shows how r remains constant while $\beta > \epsilon_\beta$ or $\alpha > \epsilon_\alpha$.
- Lastly, when it is safe to execute the final approach, the chaser enters the safety sphere under the control of the final controllers as shown by line segment D.

The simulation stops when the docking sequence is complete ($r = 0.4$ m as was explained in section 12.2).

13.3.1 Malan's EKF

As soon as enough lights have been identified, Malan's EKF will be updated. This happens for the first time at $t = 3.2$ seconds, resulting in the large corrections seen at that time.

For convenience, the figures in this section have been divided into 3 regions: The far, close and final controllers are active in the gray region on the left, the white region in the middle and the gray region on the right, respectively.

Figures 13.3 and 13.5 gives the target's position along the camera's z- and y-axis, respectively. From these figures it can be seen that the target's estimated position corresponds well with its actual position.

The target's velocity along the camera's z- and y-axis is given by figures 13.4 and 13.6, respectively. Although noisy, the estimated velocity tracks the actual velocity very well. In figure 13.4 it can be seen that v_z clearly steps from -0.02 to 0.0 to -0.02 m.s^{-1} . This is because for $t > 40$ seconds $\alpha \approx 0$ (see figure 13.10), so that $v_z \approx v_r$.

Figure 13.7 shows that the target's estimated and actual orientation relative to the chaser correlates well. The relative orientation converges to π rad as when the docking mechanisms are facing each other, the relative orientation must be π rad.

The target's estimated and actual angular rate about its z-axis relative the the chaser's angular rate around its z-axis compares well. Figure 13.8 shows that although noisy, the estimated angular rate clearly follows the actual angular rate.

Table 13.1 lists the RMS errors between the actual and estimated values.

Parameter	RMS Error	Unit
z	0.0046	m
y	0.0073	m
v_z	0.0063	m.s^{-1}
v_y	0.0123	m.s^{-1}
ψ	0.0129	rad
ω_{T_z/C_z}	0.0236	rad.s^{-1}

Table 13.1: The RMS errors of Malan's EKF.

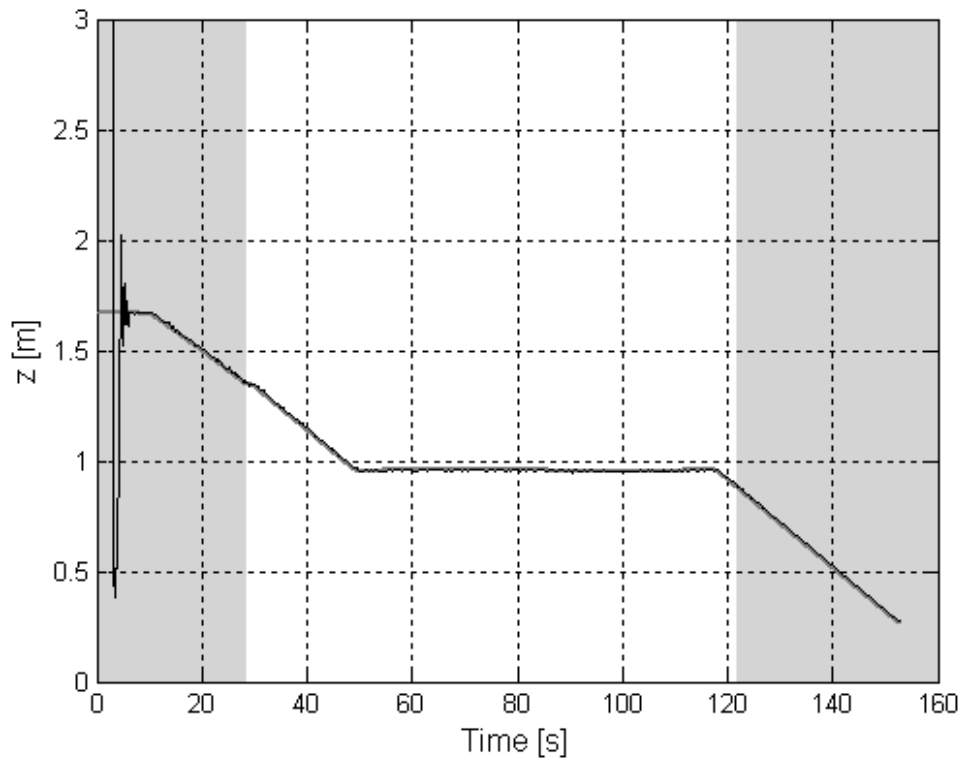


Figure 13.3: Target's estimated (black) and actual (gray) position along the camera's z-axis.

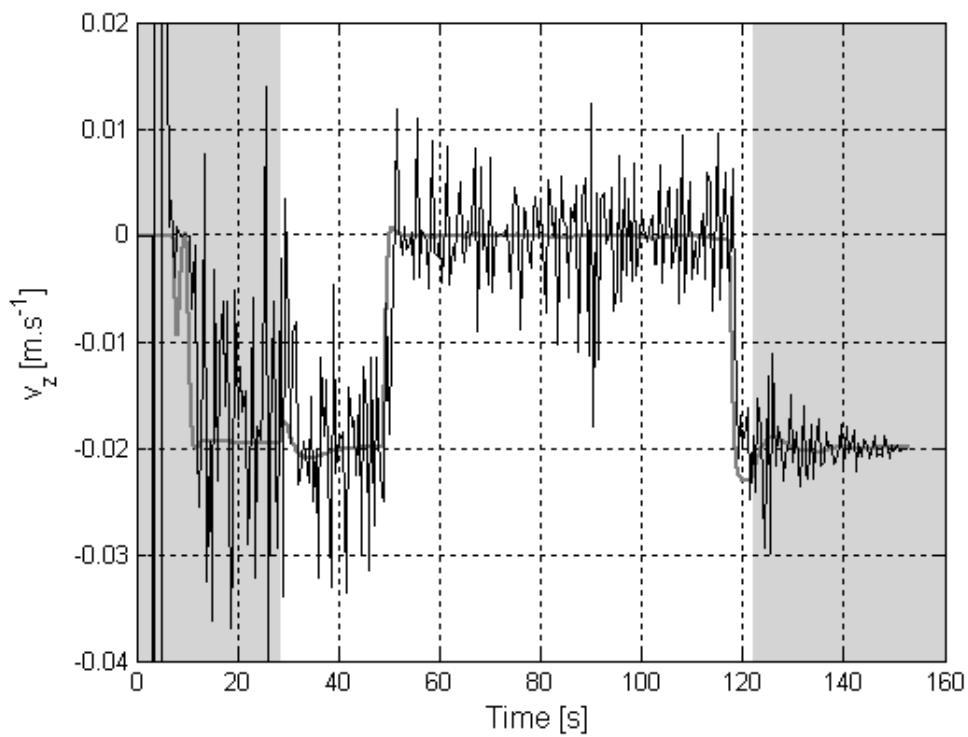


Figure 13.4: Target's estimated (black) and actual (gray) velocity along the camera's z-axis.

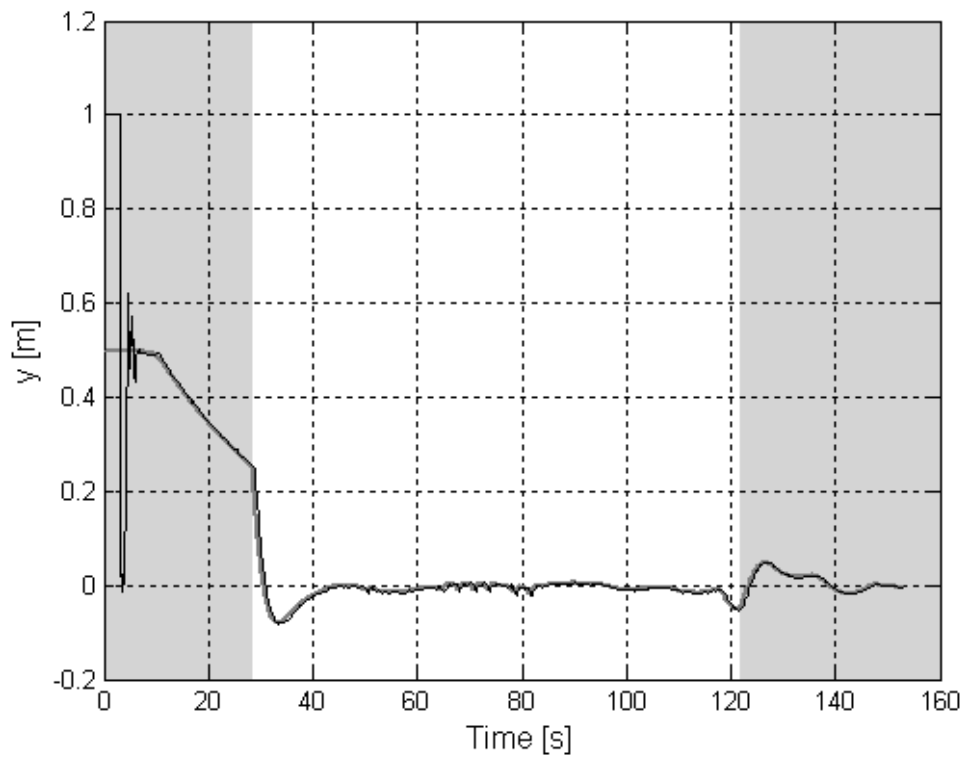


Figure 13.5: Target's estimated (black) and actual (gray) position along the camera's y-axis.

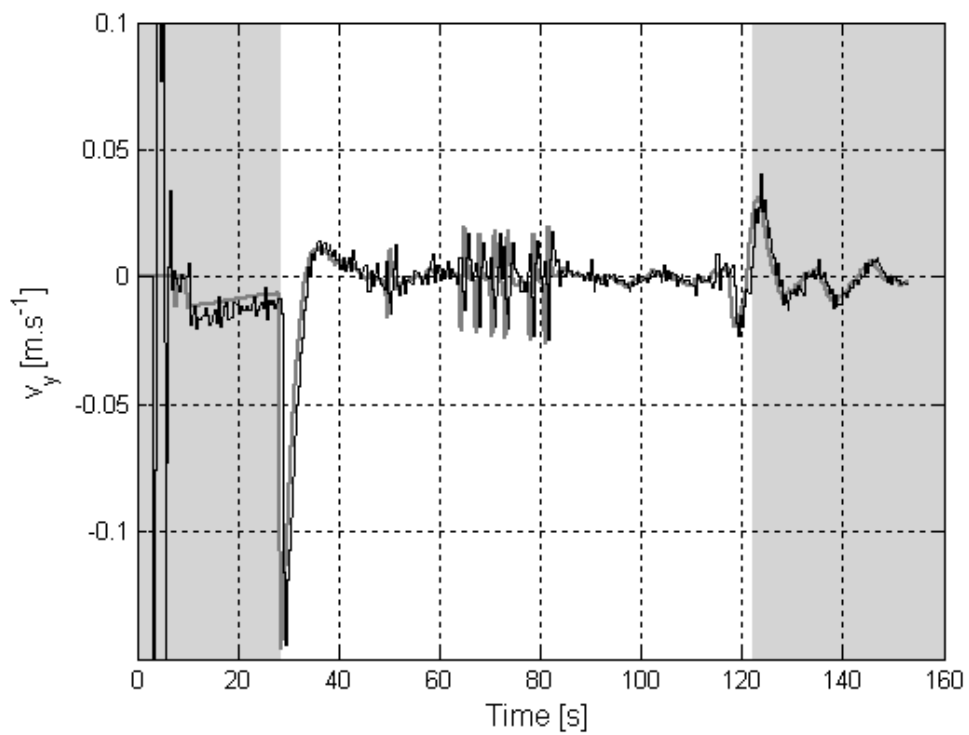


Figure 13.6: Target's estimated (black) and actual (gray) velocity along the camera's y-axis.

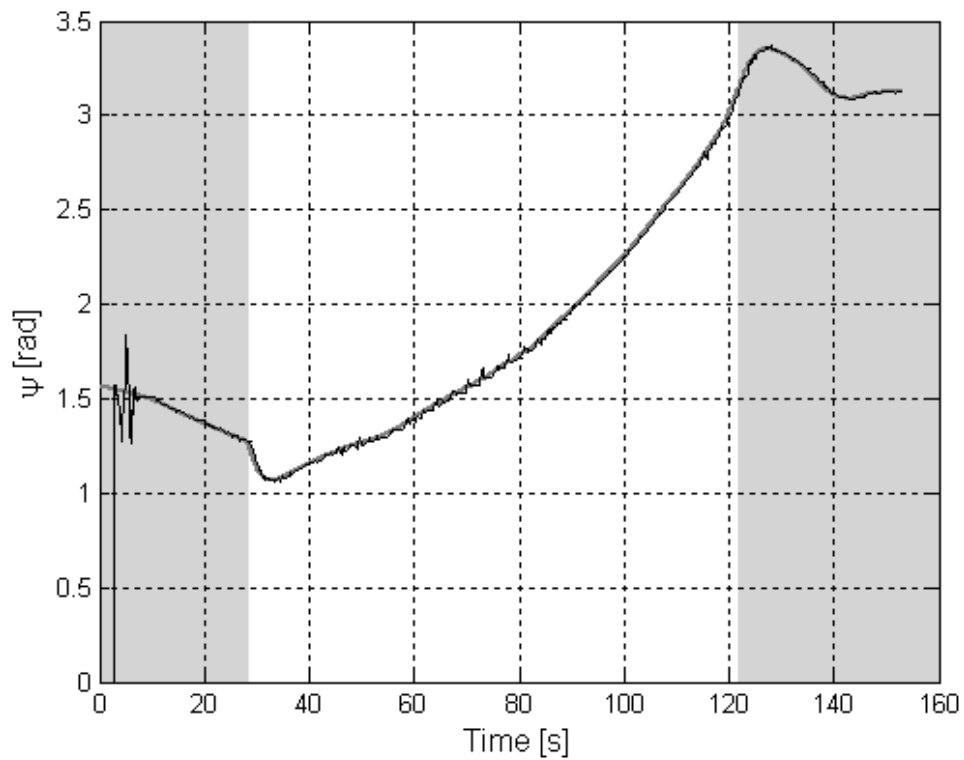


Figure 13.7: Target's estimated (black) and actual (gray) orientation with respect to the chaser.

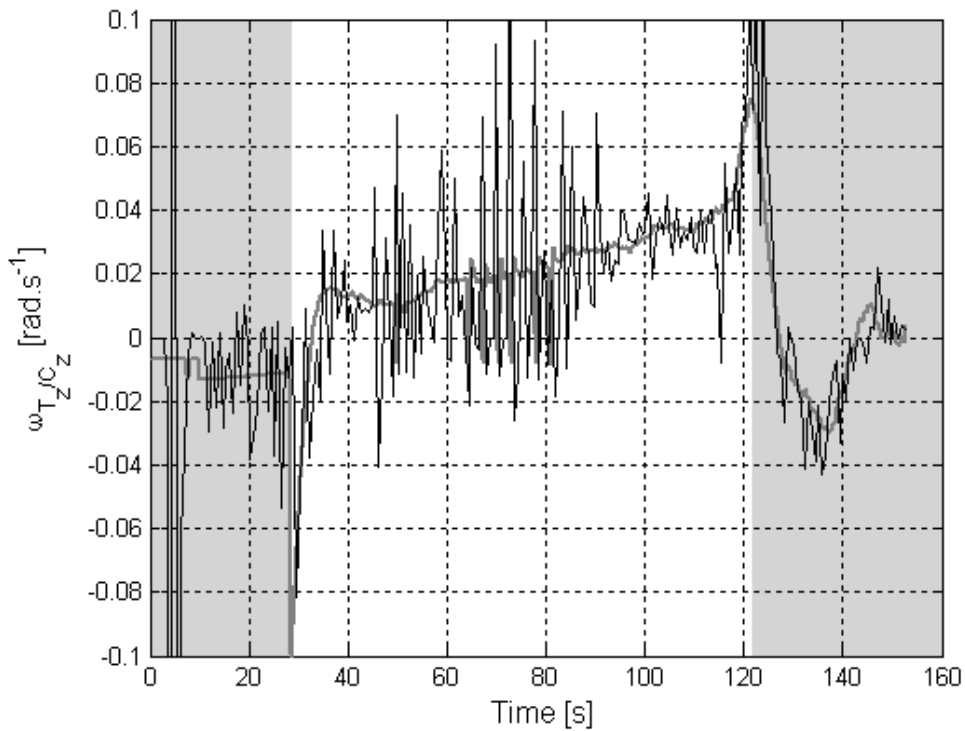


Figure 13.8: Target's estimated (black) and actual (gray) angular velocity with respect to the chaser's angular velocity.

13.3.2 System EKF

Once Malan's EKF has converged, the system EKF can also be updated. This happens for the first time at $t = 7.3$ seconds, resulting in the large corrections seen at that time.

Again, note that the figures in this section are divided into 3 regions: The far, close and final controllers are active in the gray region on the left, the white region in the middle and the gray region on the right, respectively.

The figures in this section shows that the estimated and actual parameters correlates very well.

In figure 13.9 it be seen how the distance r is first decreased at a constant velocity, after which r is kept constant while $\beta > \epsilon_\beta$ or $\alpha > \epsilon_\alpha$. When it is safe to dock, the distance r is again decreased at a constant velocity until it reaches its minimum value of 0.4 m.

As can be seen in figure 13.10, the angle α is first slowly regulated to zero under the control of the far controllers. Then, when the close and final controllers take over, the angle α is regulated more aggressively to zero.

Figure 13.11 shows that the angle β increases at a constant rate while the far controllers are active. This is because the far controllers does not regulate β to zero so that β increases at a rate ω_{T_z} . Then, under the control of the close controllers, the angle β is slowly regulated to zero. Thereafter, β is aggressively regulated to zero by the final controllers.

Lastly, figure 13.12 shows how the target's estimated angular velocity, ω_{T_z} converges to the actual value of $-0.007 \text{ rad.s}^{-1}$.

Parameter	RMS Error	Unit
r	0.0062	m
α	0.0061	rad
β	0.0086	rad
ω_{T_z}	8.0446×10^{-4}	rad.s^{-1}

Table 13.2: The RMS errors of system EKF.

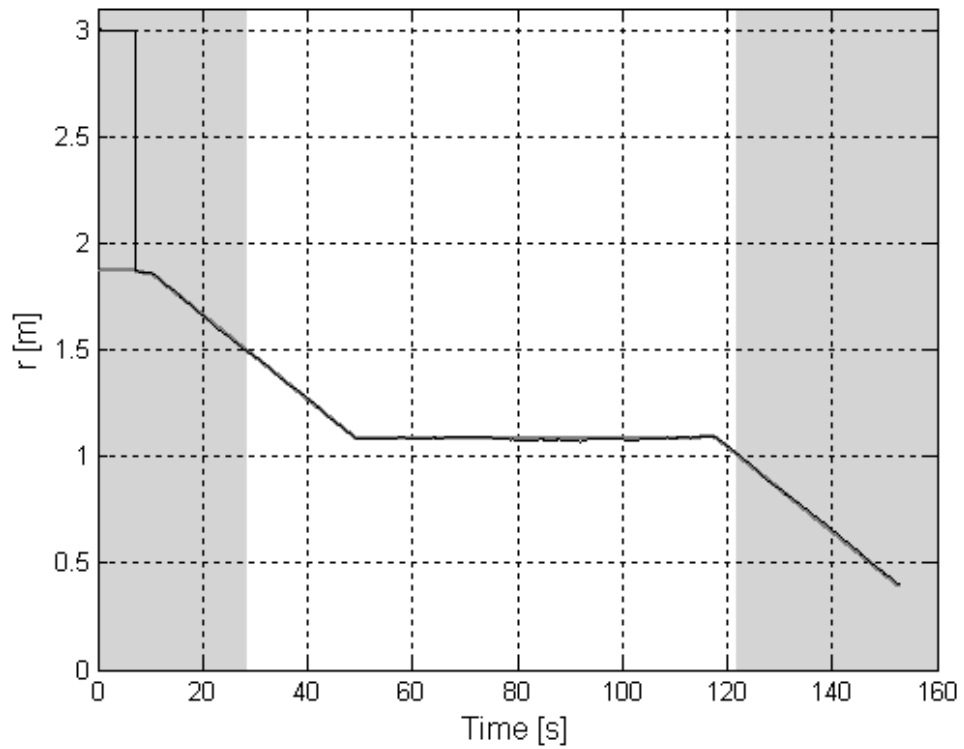


Figure 13.9: The estimated (black) and actual (gray) distance r .

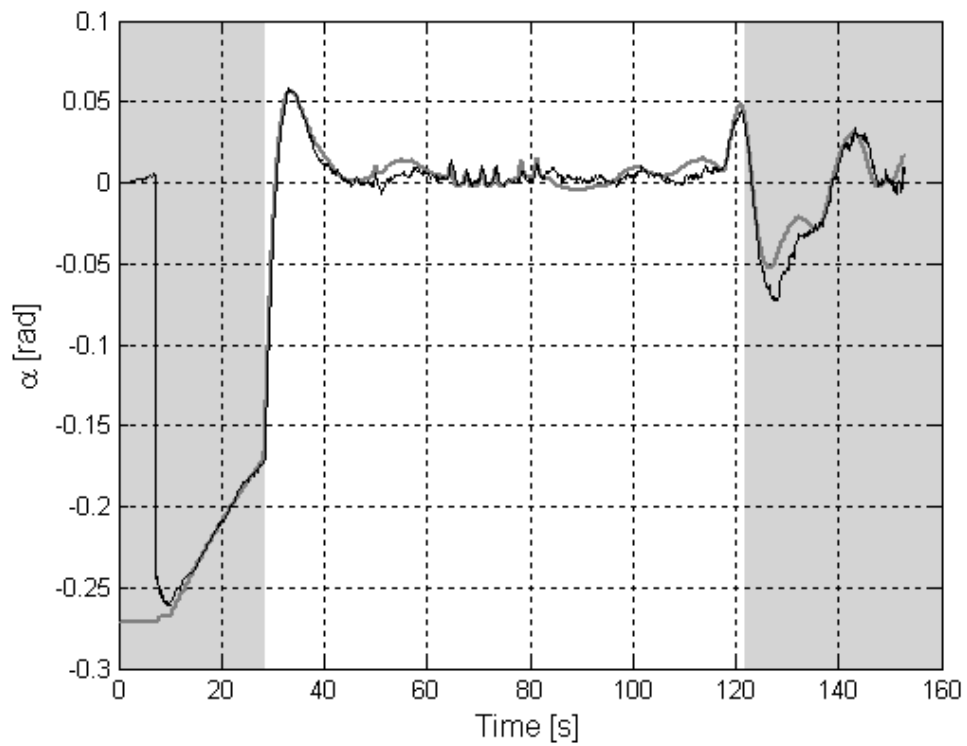


Figure 13.10: The estimated (black) and actual (gray) angle α .

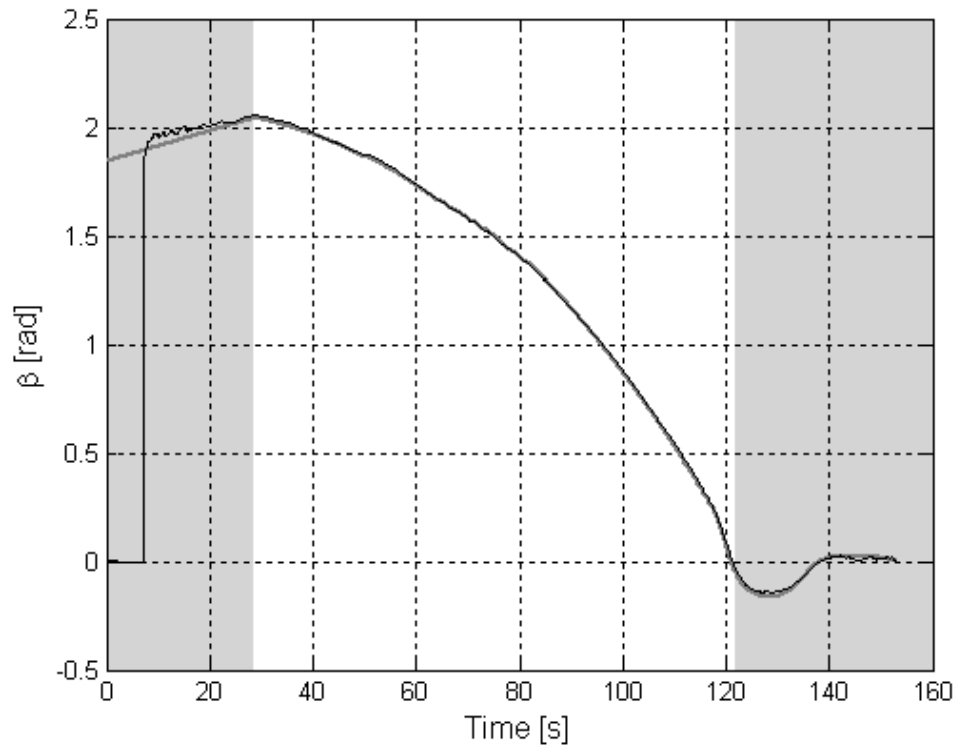


Figure 13.11: The estimated (black) and actual (gray) angle β .

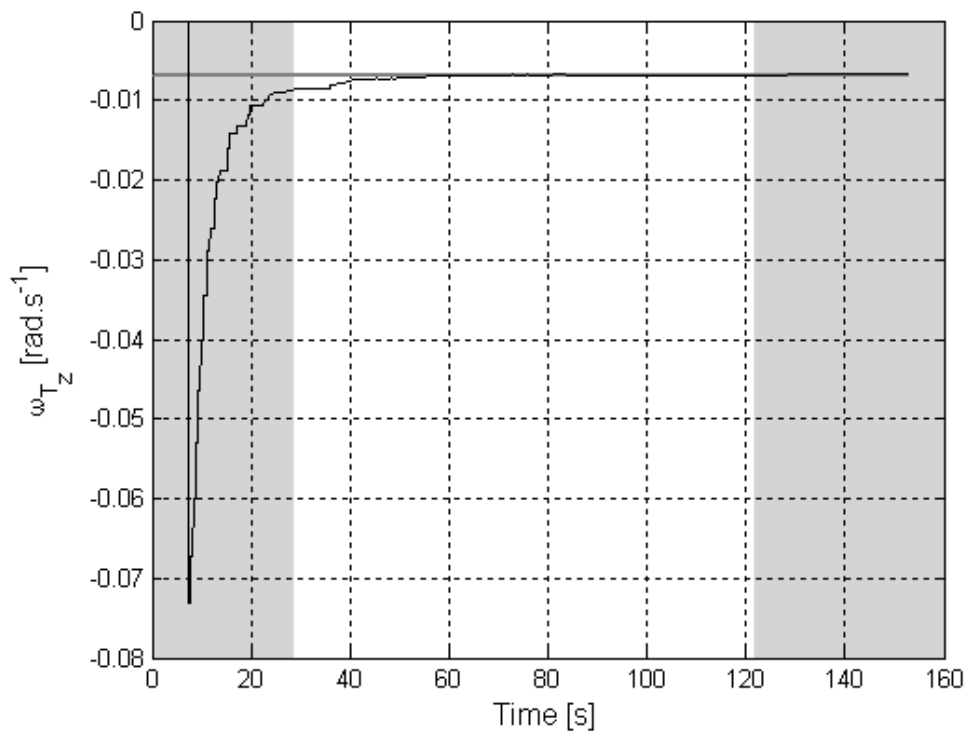


Figure 13.12: Target's estimated (black) and actual (gray) angular rate around its own z-axis.

13.4 Conclusion

This chapter showed the results of a simulation for a specific test case. Several tests were run and all of them was successful.

The simulation was very successful in the sense that the system was modeled accurately enough so that it worked in practice without any major changes apart from the interface functions.

Chapter 14

Practical Results

This chapter presents the results from a practical test.

14.1 The Test Setup

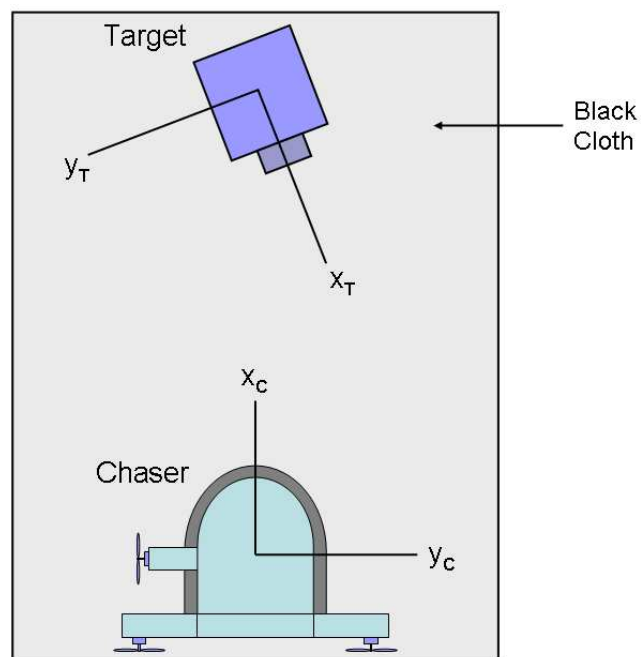


Figure 14.1: The test setup.

Figure 14.1 shows the test setup. The test had to be executed on a thin black cloth as the target's lights reflect from the floor. In this test the chaser and target were placed so that the initial values of r , α and β should be approximately 1.8 m, 0 rad and 0.175 rad (10°), respectively. The target was stationary so that ω_{T_z} is 0 $\text{rad}\cdot\text{s}^{-1}$.

The estimated values of Malan's EKF and the system EKF for this practical test is given in sections 14.1.1 and 14.1.2, respectively. As in chapter 13, the figures are divided into 3 regions: The far, close and final controllers are active in the gray region on the left, the white region in the middle and the gray region on the right, respectively.

14.1.1 Malan's EKF

Once all the lights are identified, Malan's EKF can be updated. This happens for the first time at $t = 10$ seconds resulting in the large corrections seen at that time.

Figure 14.2 shows the target's estimated position along the camera's z-axis. Initially it is corrected to the expected value of 1.67 m ($1.8 - x_{CAM}^{CHR}$, where $x_{CAM}^{CHR} = 0.13$ m). The relationship between \bar{z} and \bar{r} is given by (refer to figure 11.1):

$$\begin{aligned}\bar{z} &= \bar{r} \cos(\bar{\alpha}) - x_{CAM}^{CHR} \\ &= \bar{r} \cos(\bar{\alpha}) - 0.13\end{aligned}$$

As the estimated angle $\bar{\alpha}$ is regulated to zero (refer to figure 14.9):

$$|\bar{\alpha}| < 0.06\text{rad} = 3.44^\circ$$

So that:

$$\cos(\bar{\alpha}) > 0.9982$$

Therefore:

$$\bar{z} \approx \bar{r} - 0.13$$

This can be seen by comparing figures 14.2 and 14.8.

The target's average estimated velocity along the camera's z-axis is -0.0162 m.s^{-1} . As the estimated angle $\bar{\alpha}$ is closely regulated to zero as shown before:

$$\bar{v}_Z \approx v_{r(ref)}$$

It is therefore expected that \bar{v}_Z should be -0.02 m.s^{-1} , the radial velocity reference. The deviation can be attributed to the black cloth used as this cloth impedes the hovercraft's movement. Refer to figure 14.3.

Because the estimated angle $\bar{\alpha}$ is regulated to zero, the target remains centred in the image so that the target's estimated position along the camera's y-axis, \bar{y} , should be small. This can also be seen from the relationship (refer to figure 11.1):

$$\bar{y} = \bar{r} \sin(\bar{\alpha})$$

As $|\bar{\alpha}| < 3.44^\circ$:

$$\begin{aligned}\bar{y} &\leq \bar{r}(0.06) \\ &\leq (1.8)(0.06) \\ &\leq 0.108\end{aligned}$$

This can be seen in figure 14.4.

Figure 14.5 shows the target's estimated velocity along the camera's y-axis. As the estimated angle $\bar{\alpha}$ is slowly regulated to zero, \bar{v}_Y remains close to zero.

From figure 14.1 it can be seen that the orientation of the target's body axis relative to the chaser's body axis should be roughly π rad (180°). This is reflected in figure 14.6.

Lastly, figure 14.7 shows the target's estimated angular velocity with respect to the chaser's angular velocity, $\bar{\omega}_{T_Z/C_Z}$. The small variations seen, $|\bar{\omega}_{T_Z/C_Z}| \leq 0.436 \text{ rad.s}^{-1} = 2.49^\circ.\text{s}^{-1}$, are the result of the angles α and β being slowly regulated to zero. This can be seen by combining the following equations:

$$\begin{aligned}\dot{\alpha} &= \omega_{C_Z} - \omega_{C_T} \\ \dot{\beta} &= \omega_{C_T} - \omega_{T_Z} \\ \omega_{T_Z/C_Z} &= \omega_{T_Z} - \omega_{C_Z}\end{aligned}$$

Remembering that ω_{T_Z} is zero, as the target is stationary in this test, results in:

$$\omega_{T_Z/C_Z} = -(\dot{\alpha} + \dot{\beta})$$

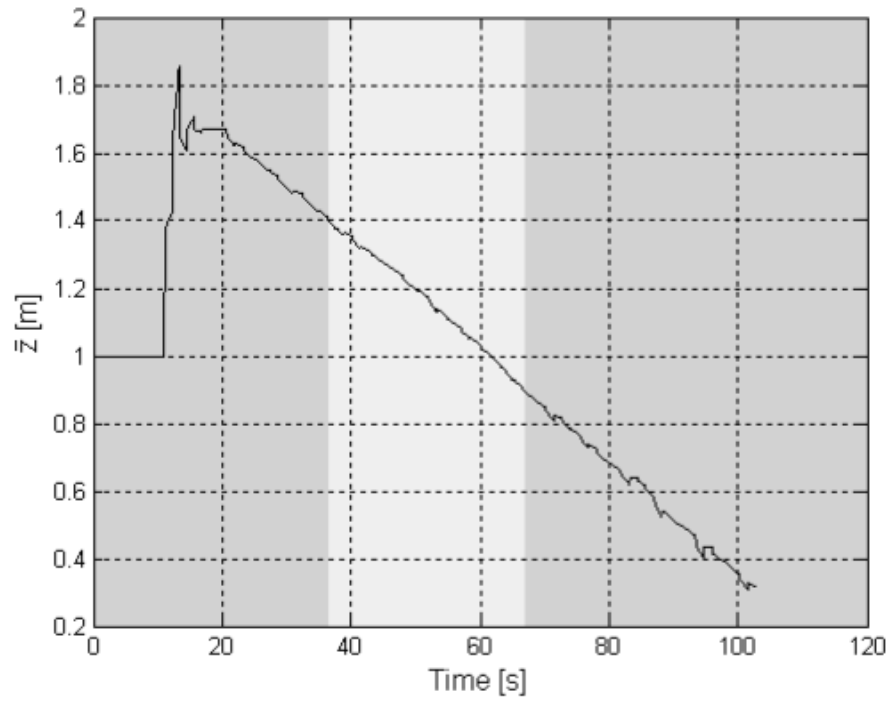


Figure 14.2: The target's estimated position along the camera's z-axis.

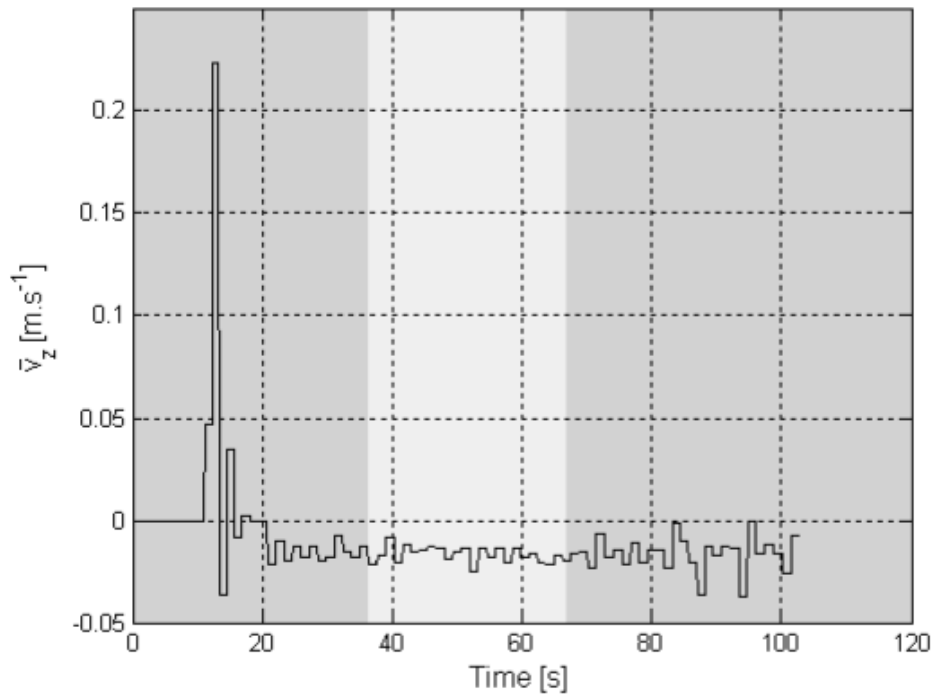


Figure 14.3: The target's estimated velocity along the camera's z-axis.

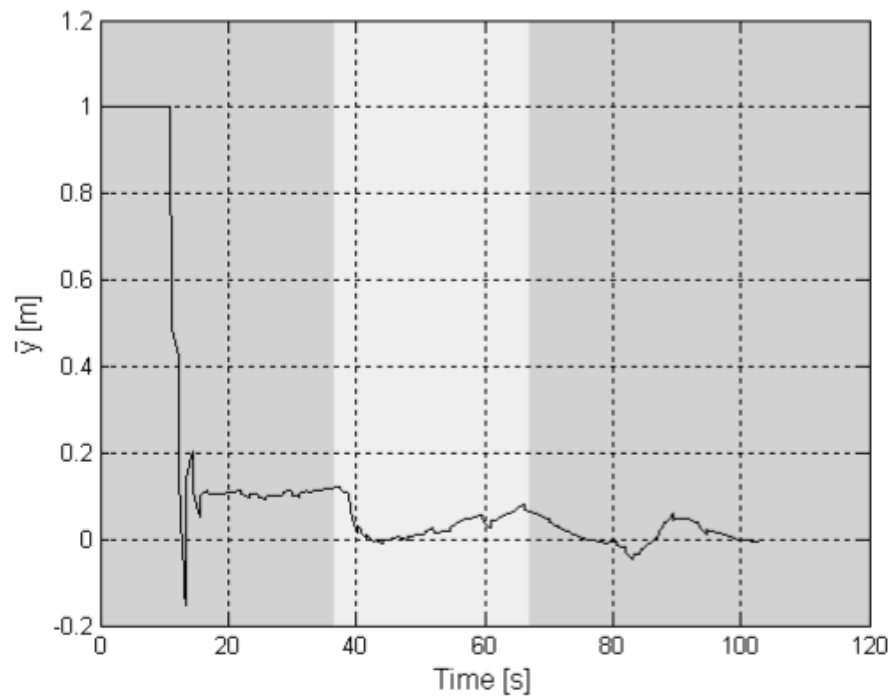


Figure 14.4: The target's estimated position along the camera's y-axis.

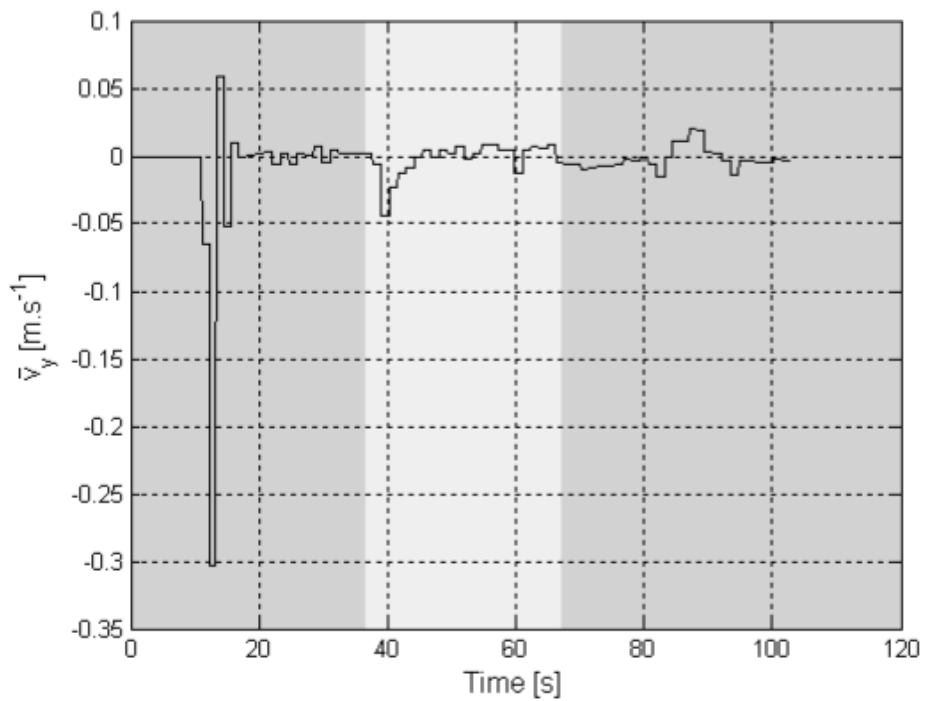


Figure 14.5: The target's estimated velocity along the camera's y-axis.

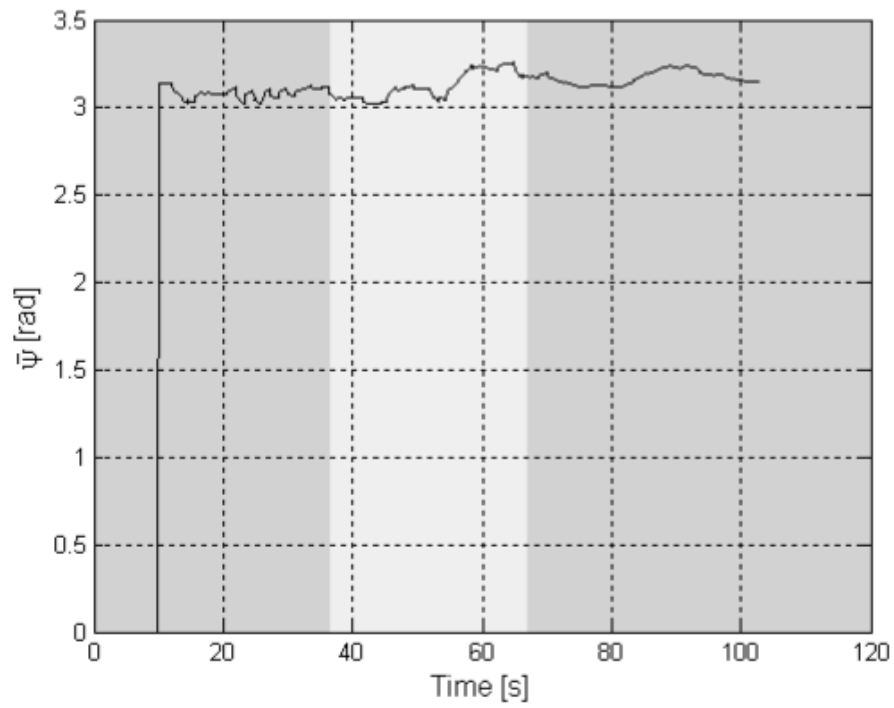


Figure 14.6: The target's estimated orientation with respect to the chaser.

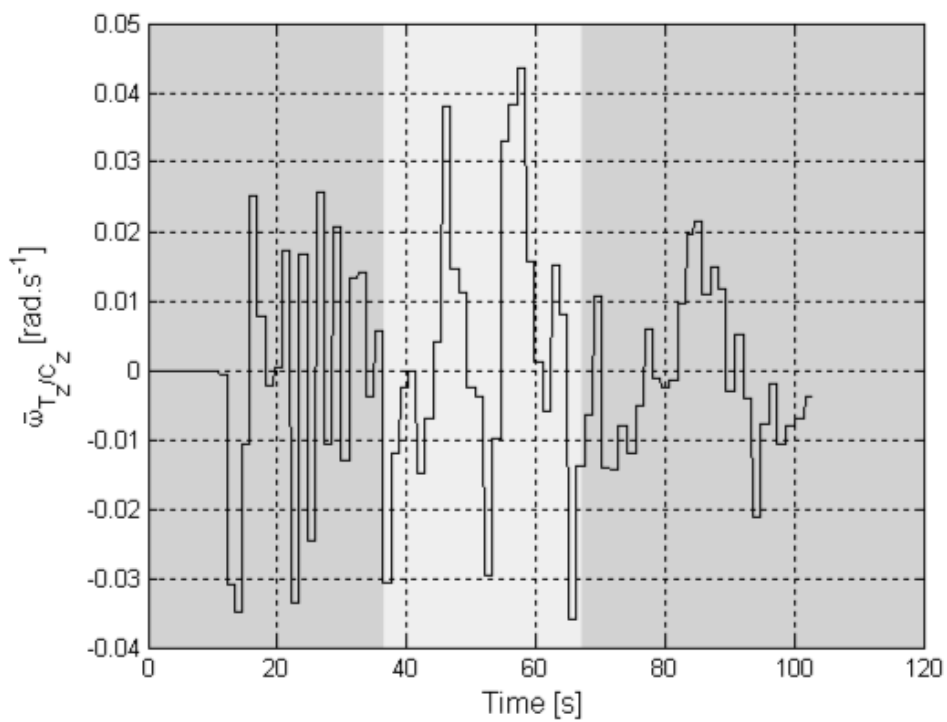


Figure 14.7: Target's estimated angular velocity with respect to the chaser's angular velocity.

14.1.2 System EKF

Once Malan's EKF converged, the system EKF can be updated. This happens for the first time at $t = 17$ seconds resulting in the big corrections seen at that time. Note that as soon as the system EKF has converged, the controllers are activated.

Figure 14.8 shows the estimated distance \bar{r} . The distance decreases linearly, from the expected initial value of 1.8 m to the final distance of 0.4 m, implying a constant velocity. From this graph, the average velocity is calculated as -0.0166 m.s^{-1} . This is 17% less than the reference velocity of -0.02 m.s^{-1} and this deviation is attributed to the black cloth used to prevent the lights from reflecting off the ground. (Although the cloth was tensioned, it still wrinkles as the hovercraft moves across it, impeding the hovercraft's movement.)

The far controllers initially struggle to regulate the estimated angle $\bar{\alpha}$ to zero as seen in figure 14.9. It is expected that there should be a tracking error when the far controller for $\bar{\alpha}$ is active as it is only a proportional controller. Thereafter, the estimated angle $\bar{\alpha}$ is aggressively regulated to zero by the close and final controllers so that the magnitude of the estimated angle $|\bar{\alpha}|$ remains within 0.04 rad (2.29°).

Initially, the estimated angle $\bar{\beta}$ remains fairly unchanged as it is not regulated by the far controllers. It is then regulated to zero under the control of the close and final controllers as shown in figure 14.10. The final oscillation seen in the estimated angle $\bar{\beta}$ is also the result of the black cloth used. As stated earlier, the black cloth wrinkles as the hovercraft moves over it. Therefore the slow integrator in the controller for the angle β integrates up to compensate. Once the angle β overshoots, the integrator has to integrate up in the opposite direction to compensate in the opposite direction. This causes the oscillation seen.

From figure 14.11 it can be seen that the estimated $\bar{\omega}_{T_z}$ converges to $0.00038 \text{ rad.s}^{-1}$ and is very close to the correct value of 0.0 rad.s^{-1} .

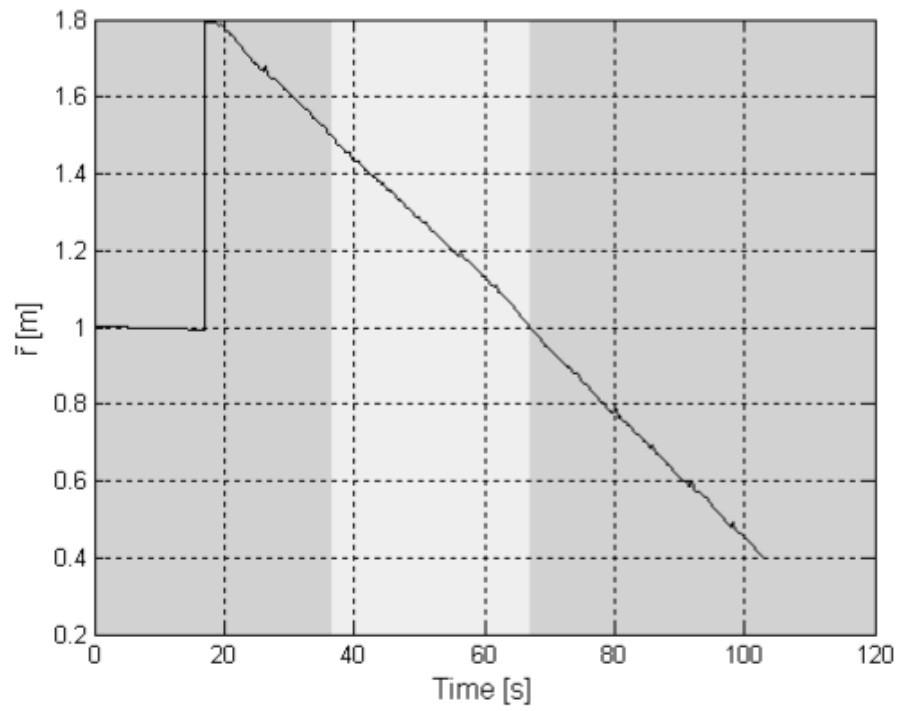


Figure 14.8: The estimated distance r .

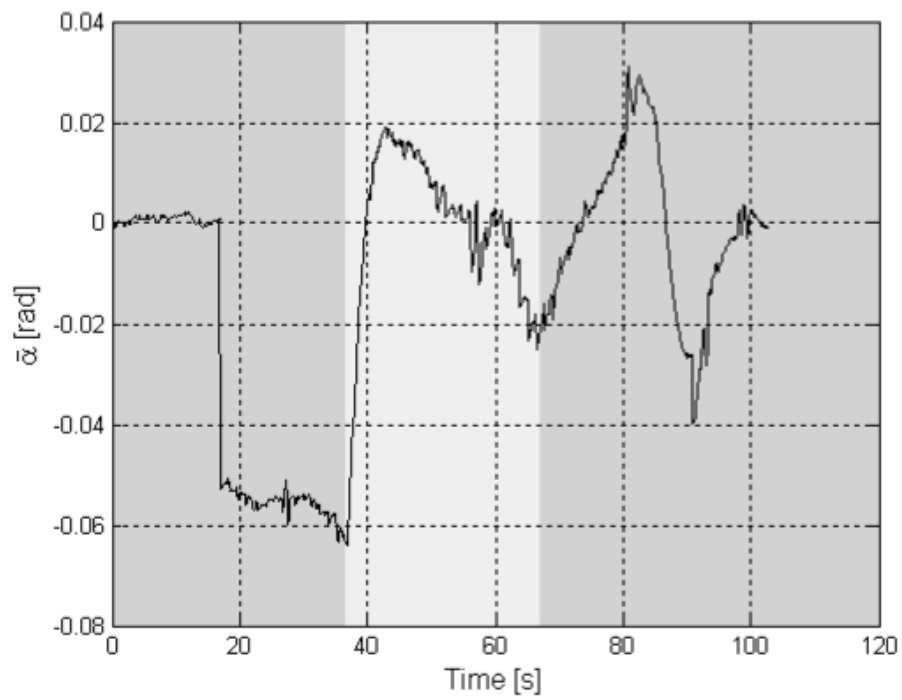


Figure 14.9: The estimated angle α .

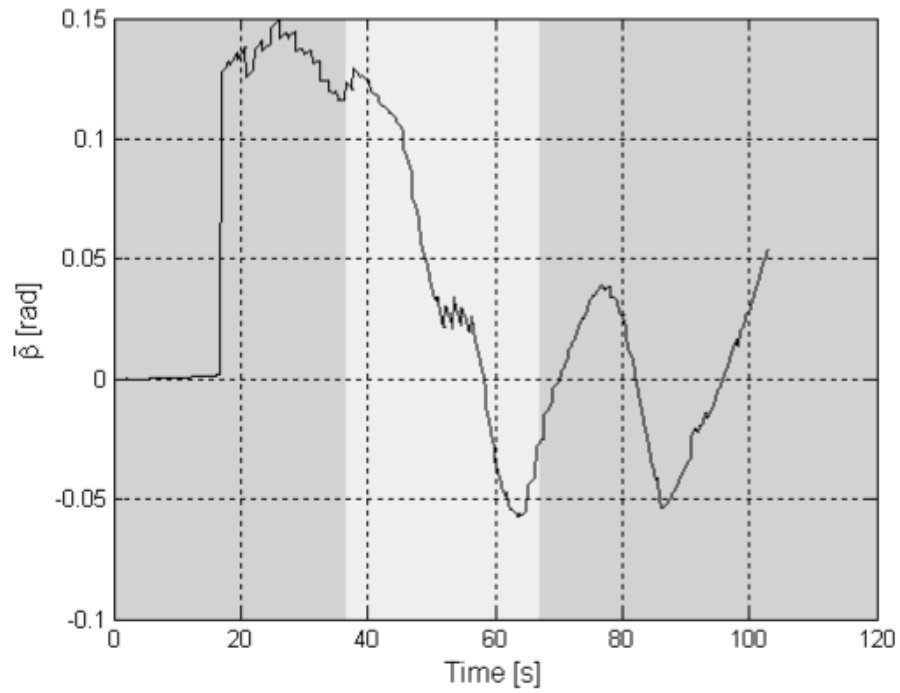


Figure 14.10: The estimated angle β .

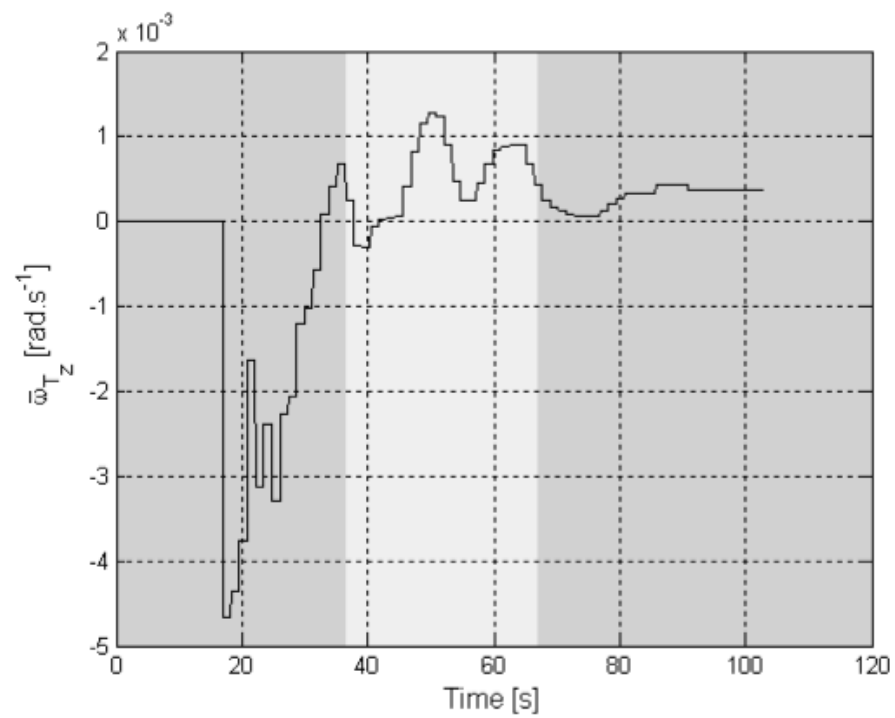


Figure 14.11: Target's estimated angular rate around its own z-axis.

14.2 Conclusion

This test was very successful. All the lights were uniquely identified, the EKFs converged and the docking sequence executed successfully.

Initially, the idea was that the practical test should imitate the scenario from the simulation. Sadly this was not possible. Apart from that the fact that the hovercraft is not such an ideal platform as seen in chapter 10, the use of a black cloth to prevent reflections of the target's lights in the floor impeded the hovercraft's movement. Although the cloth is very thin and was tensioned, it still wrinkled as the hovercraft moved over it. This wrinkling impeded the hovercraft's movement especially along the hovercraft's y-axis as there is only one side fan compared to the 2 fans along the hovercraft's x-axis. As result the force along the hovercraft's x-axis is enough to move over this wrinkles (typically causing the hovercraft to "jump" forward), but this is not true for the y-axis. Therefore the manoeuvre from the simulation could not be executed.

Due to time limitations it was not possible to create an setup to take external measurements to determine the EKFs' accuracies. But, if one takes into account the size of the hole in the target (see figure D.5) and consider the distance to the tip of the pin (see figure D.4) from the hovercraft's body axis origin, an estimate of the accuracies of the estimated angles $\bar{\alpha}$ and $\bar{\beta}$ can be made. The estimated accuracies are:

- $|\alpha| < 0.0322 \text{ rad (1.8476}^\circ\text{)}$
- $|\beta| < 0.0713 \text{ rad (4.0856}^\circ\text{)}$

The accuracy of ω_{T_z} can be seen in figure 14.11 as it must be zero.

Chapter 15

Conclusion and Recommendations

Autonomous rendezvous and docking are seen as enabling technologies. It allows, among others, the in-orbit construction of larger space platforms and the in-orbit servicing of space vehicles.

Although the docking can be done by a remote human operator, autonomous docking is preferred as it does not rely on human capabilities and a communications link to the ground.

A prerequisite for autonomous docking is the ability to estimate the relative position, velocity, angular rate and orientation of the target satellite with respect to the chaser satellite. This is done using a single camera (i.e. monocular vision) as it is considered to be the best overall sensor in terms of performance, mass and power consumption.

The goal of this thesis is therefore to demonstrate (in simulation and emulation) the concept of docking two satellites autonomously using monocular camera vision. This entails the research, design and implementation of a docking algorithm, camera sensor system and the creation of a test platform for emulation.

15.1 Conclusions

A low impact docking sequence was designed as it has the advantage of requiring smaller and lighter docking mechanisms. The docking sequence is divided into 3 phases (according to the distance between the satellites) as it provides a logical way to divide the problem into subproblems. According to the current phase, a guidance system schedules the necessary controllers to execute the manoeuvres required to complete the docking sequence. Although the docking sequence is described in 2 dimensions, it is designed as such that the principles apply directly in 3 dimensions. This docking sequence was successfully demonstrated in both simulation and emulation environments.

As all the other sensors, the camera sensor has to be calibrated and the lens' radial distortion must be compensated for. A calibration procedure was designed and used to successfully calibrate 3 cameras in the ESL. During the calibration process, the camera has to be mounted in front of a test pattern. To do this, a camera clamp was designed. This clamp was also designed in such a way that it can accommodate lenses of varying diameters.

It was decided to use a model hovercraft to emulate the chaser satellite in 2 dimensions as it moves relatively frictionless. The hovercraft is equipped with the necessary hardware for

the emulation. This includes a camera, onboard computer, angular rate gyro, velocity sensor and wireless link. A motor driver board was also made to control the hovercraft's fans. The hovercraft's original lift fan was also replaced as it could not take the strain of the extra load.

The hovercraft's velocity and angular rate is controlled by a set of inner loop controllers. These controllers were designed using a model of the hovercraft, the actuators and the sensors. (All the sensors and actuators also had to be calibrated.)

An imitation of a target satellite was made and equipped with light markers. After a literature study on the different methods used to extract the necessary data from a sequence of images was completed, it was decided to make use of light markers. By observing the position of the target's lights in the image the necessary relative parameters can be estimated. The approach followed was the most compatible with the current hardware setup.

One requirement of this approach is the ability to uniquely identify each light marker in the image. An algorithm was designed to uniquely and reliably identify and track the target's lights' centroids in the presence of noise, e.g. stars. The identification and tracking of the lights in the image are aided by a light model. This light model was designed in such a way that it takes the lights' radiation patterns into account. The light model also acts as a virtual camera in the simulation of the docking sequence. Based on the outcome of the simulation and emulation, this identification and tracking algorithm look promising.

A region growing approach is normally used to determine the centroids. The main disadvantage of this approach is that it reads each pixel up to 4 times. In order to improve the camera sensor's bandwidth a different centroid determination algorithm was designed. Apart from taking the camera's memory structure into account, the main advantage of this algorithm is that it only reads each pixel once. It was shown that this algorithm is up to 4 times faster than the region growing algorithm (depending on the mass of a centroid).

By using a modified version of the EKF designed by Malan [20] the target's position, velocity, angular rate and orientation in the camera axis can be estimated from the positions of the target's lights in the image. The EKF designed by Malan has 13 states as it estimates all these quantities in 3 dimensions. As the emulation is only done in 2 dimensions, most of these states are not required. His filter was therefore customised for this application and the modified EKF only has 6 states, reducing the computational complexity.

The output of the modified version of Malan's EKF is combined with measurements from a velocity sensor and a rate gyro in another EKF, the system EKF. (Due to mechanical vibrations and the tilting of the hovercraft the accelerometers, that were originally used, did not work satisfactory and had to be replaced with a velocity sensor.) The system EKF's information is then used by the guidance system and outer loop controllers to execute the docking sequence. Three sets of outer loop controllers were designed and each set is scheduled by the guidance system depending on the current docking stage.

A simulation was written in C to test the concepts in this thesis. A layered software topology was used to allow the simulation software to be easily ported to the hardware. This simulation was very successful in the sense that the system (hovercraft, inner loop controllers, actuators, camera, barrel distortion, sensors, sensor noise, light radiation patterns etc.) was modeled accurately enough so that the simulation software was ported to the hardware without any major changes.

An emulation of the docking sequence was also done using the modified model hovercraft and the imitation of the target satellite. The hovercraft successfully identified all the lights and also successfully completed the docking sequence.

In conclusion, this thesis successfully demonstrated the autonomous docking of a satellite pair using monocular vision in both simulation and emulation conditions. This shows that the ideas used in this thesis are promising for satellite implementation.

15.2 Recommendations

Here is a short list of recommendations:

- It was found that the hovercraft is not such an ideal platform as was anticipated. Even though it does move relatively frictionless, it has little resistance against moments that causes the body to tilt. (Discussed in chapter 10.) For future research it would be advised to use a more suitable platform such as the one in figure 15.1 used by Romano [27].

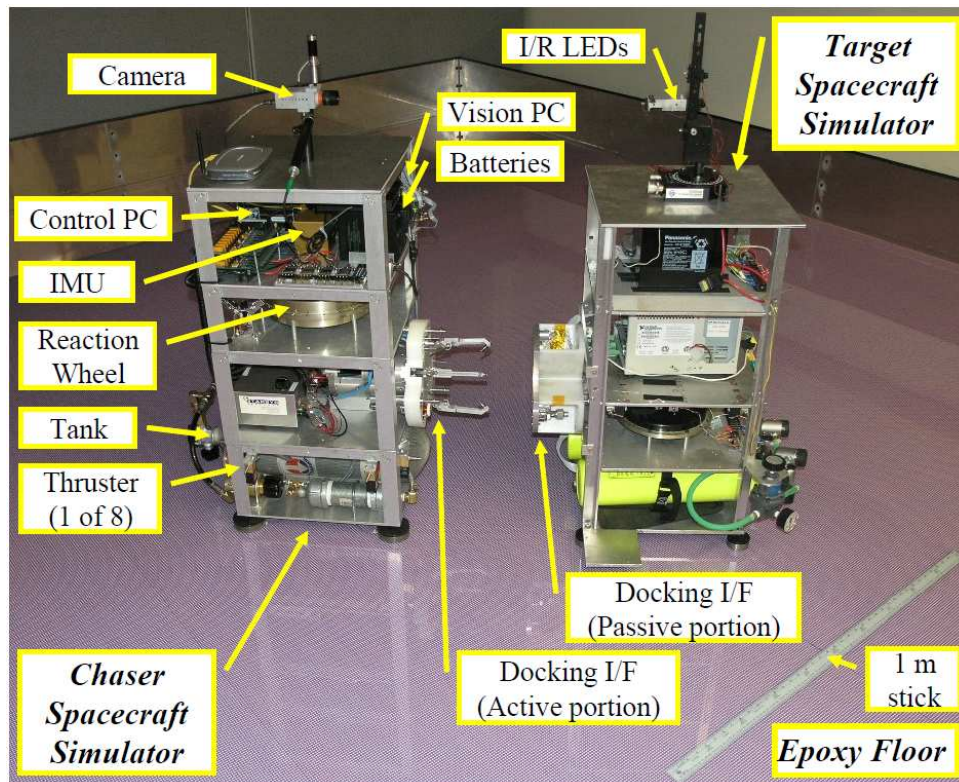


Figure 15.1: Autonomous docking test bed used by Romano. (Picture taken from [27].)

- Due to time restrictions, the accuracies of Malan's EKF and the system EKF could not be verified by external measurements. This can still be done.

- The ideas in this thesis provides a stepping stone for optimal path planning (in terms of fuel usage). An optimal path can, for example, be defined as a set of parametric equations:

$$\begin{aligned}\alpha &= f_{\alpha}(r) \\ \beta &= f_{\beta}(r) \\ v_r &= f_{v_r}(r)\end{aligned}$$

It must also be possible to adjust this path in real time to accommodate for unplanned deviations.

Bibliography

- [1] Bijker, J., *Development of an Attitude Reference System for an Airship*, Masters Thesis, Stellenbosch University, 2006.
- [2] Cropp, A., Palmer, P., and Underwood, C.I., *Pose Estimation Of Target Satellite For Proximity Operations*, 14th AIAA/USU Small Satellite Conference, Utah, Proceedings Session SSC00-III-7, 2000.
- [3] Cucchiara, R., C. Grana, A. Prati and R. Vezzani, *A Hough Transform-based method for Radial Lens Distortion Correction*, 12th International Conference on Image Analysis and Processing, Mantova, Italy, pp. 182-187, 17-19 September 2003.
- [4] Di Sotto, E., Penin, L.F., Camara, F., and Caramagno, A., *Design and Performance Assessment of Guidance Algorithms for Vision Based Rendezvous*, AIAA Guidance Navigation and Control Conference and Exhibit, Keystone, Colorado, No. AIAA 2006-6586, 21-24 August 2006.
- [5] Farid, H., and Popescu, A.C., *Blind removal of lens distortion*, Optical Society of America, Vol. 18, No. 9, 2001.
- [6] Fehse, W, *Automated Rendezvous and Docking of Spacecraft*, Cambridge Aerospace Series, Cambridge University Press, ISBN: 0521824923, 2003.
- [7] Franklin, G. F., Powell, J.D., and Workman, M., *Digital control of dynamic systems*, Third Edition, Addison Wesley Longman Inc., ISBN: 0201331535, 1997.
- [8] Furguson, P., Busse, F., Engberg, B., How, J., Tillerson, M., Pohlman, N., Richards, A., and Twiggs, R., *Formation Flying Experiments on the Orion-Emerald Mission*, AIAA Space 2001 - Conference and Exposition, Albuquerque, 28-30 August 2001.
- [9] Groenewald, S., *Development of a Rotary-Wing Test Bed for Autonomous Flight*, Masters Thesis, Stellenbosch University, 2006.
- [10] Hartley, R., and Zisserman, A., *Multiple View Geometry in computer vision*, Second Edition, Cambridge University Press, ISBN: 0521540518, 2003.
- [11] Herselman, L., *An Investigation into Underwater Navigation Accuracy with regards to Sensor Combinations and Quality*, Masters Thesis, Stellenbosch University, 2008.

- [12] Hou, H., *Modeling Inertial Sensors Errors Using Allan Variance*, Masters Thesis, University of Calgary, 2004.
- [13] Jacobs, M.J., *A low cost, high precision star sensor*, Masters Thesis, Stellenbosch University, 1995.
- [14] Kawano, I., Mokuno, M., Kasai, T., and Suzuki, T., *Result and Evaluation of Autonomous Rendezvous Docking Experiment of ETS-VII*, AIAA Guidance Navigation and Control Conference and Exhibit, Portland, Oregon, No. AIAA-1999-4073, August 1999.
- [15] Li, H., and Hartley, R., *A Non-iterative Method for Correcting Lens Distortion from Nine Point Correspondences*, 6th Workshop on Omnidirectional Vision, Beijing, China, 2005.
- [16] Lourakis, M.I.A., *A Brief Description of the Levenberg-Marquardt Algorithm Implemented by levmar*, <http://www.ics.forth.gr/lourakis/levmar/>, 11 February 2005.
- [17] Lowe, D.G., *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, Vol. 2, No. 60, pp. 91-110, 2004.
- [18] Ma, O., Crabtree, D., Carr, R., Gonthier, Y., Martin, E., and Piedboeuf, J.C., *Development and Applications of a Simulink-Based Satellite Docking Simulator with Generic Contact Dynamics Capabilities*, IAF abstracts, 34th COSPAR Scientific Assembly, The Second World Space Congress, Houston, USA, pp.U-3-02IAF, 10-19 October 2002.
- [19] Madsen, K., Nielsen, H.B., and Tingleff, O., *Methods for non-linear least squares problems (2nd Edition)*, Technical Report No. IMM-REP-2004-01, Denmark Technical University, 2004.
- [20] Malan, D.F., *3D Tracking between Satellites using Monocular Computer Vision*, Masters Thesis, Stellenbosch University, 2004.
- [21] Miller, D.P., and Wright, A., *Autonomous Spacecraft Docking Using Multi-Color Targets*, Proceedings of the Sixth Topical Meeting on Robotics, Monterey, California, February 1995.
- [22] Ojanen, H., *Automatic Correction of Lens Distortion by Using Digital Image Processing*, Rutgers University, Dept. of Mathematics technical report, July 1999.
- [23] Peebles, P.Z., *Probability, Random Variables and Random Signal Principles*, 4th Edition, McGraw-Hill, ISBN: 0073660078, 2001.
- [24] Pers, J., and Kovacic, S., *Nonparametric, Model-Based Radial Lens Distortion Correction Using Tilted Camera Assumption*, Proceedings of the Computer Vision Winter Workshop, Bad Aussee, Austria, pp. 286-295, February 2002.

- [25] Qureshi, F.Z., and Terzopoulos, D., *Cognitive Vision for Autonomous Satellite Rendezvous and Docking*, Proceedings of the IAPR Conference on Machine Vision Applications, Tsukuba Science City, Japan, pp. 314-319, 2005.
- [26] Remondino, F., and Fraser, C., *Digital Camera Calibration Methods: Considerations and Comparisons*, International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences, Dresden, Germany, Vol. XXXVI, No. part 5, 2006.
- [27] Romano, M., Friedman, D.A., and Shay, T.J., *Laboratory Experimentation of Autonomous Spacecraft Approach and Docking to a Collaborative Target*, Proceedings of the AIAA Guidance Navigation and Control Conference, Keystone, CO, August 2006.
- [28] Tsai, R.Y., *A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Metrology Using Of-the-Shelf TV Cameras and Lenses*, IEEE International Journal Robotics and Automation, Vol. 3(4), pp. 323-344, 1987.
- [29] Woffinden, D.C. and Geller, D.K., *Relative Angles-Only Navigation and Pose Estimation For Autonomous Orbital Rendezvous*, AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Keystone, Colorado, 21 - 24 August 2006.
- [30] Zhang, Z., *A Flexible New Technique for Camera Calibration*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), pp. 1330-1334, 2000.

Appendices

Appendix A

System Block Diagrams

This appendix gives an overview of the docking software. The purpose of the following block diagrams is to give a functional description of the software rather than describing the exact implementation of the software.

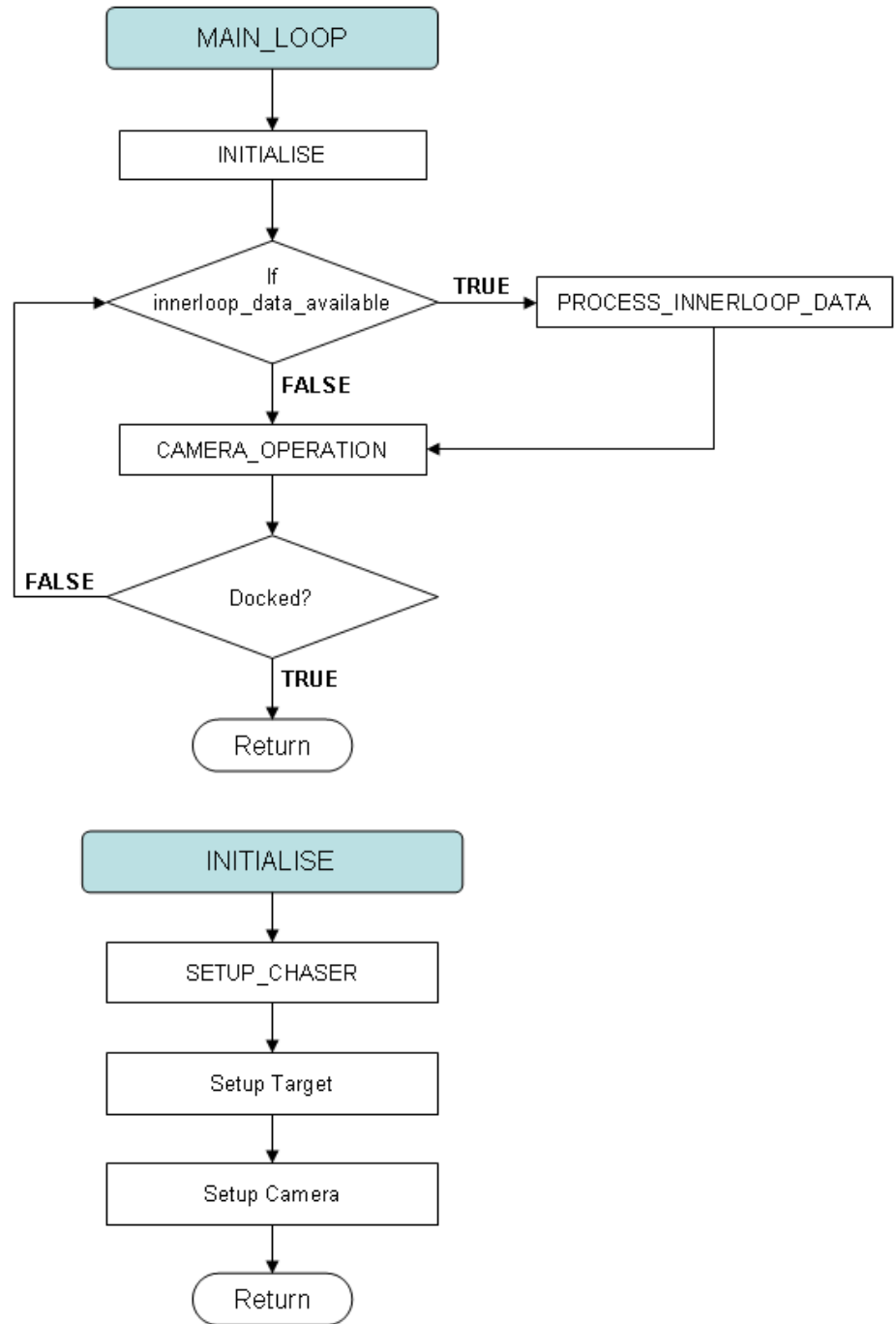


Figure A.1: Overview 1.

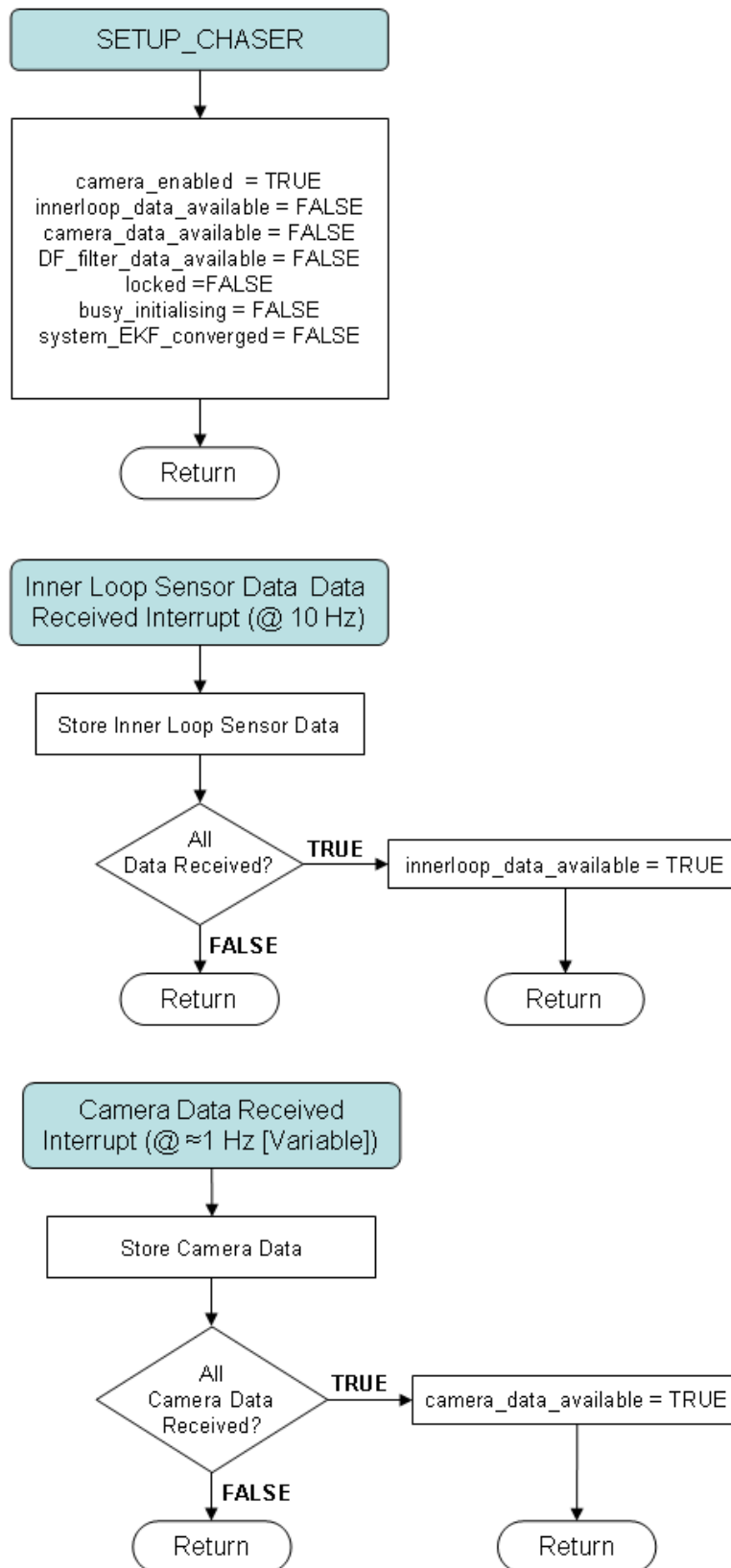


Figure A.2: Overview 2.

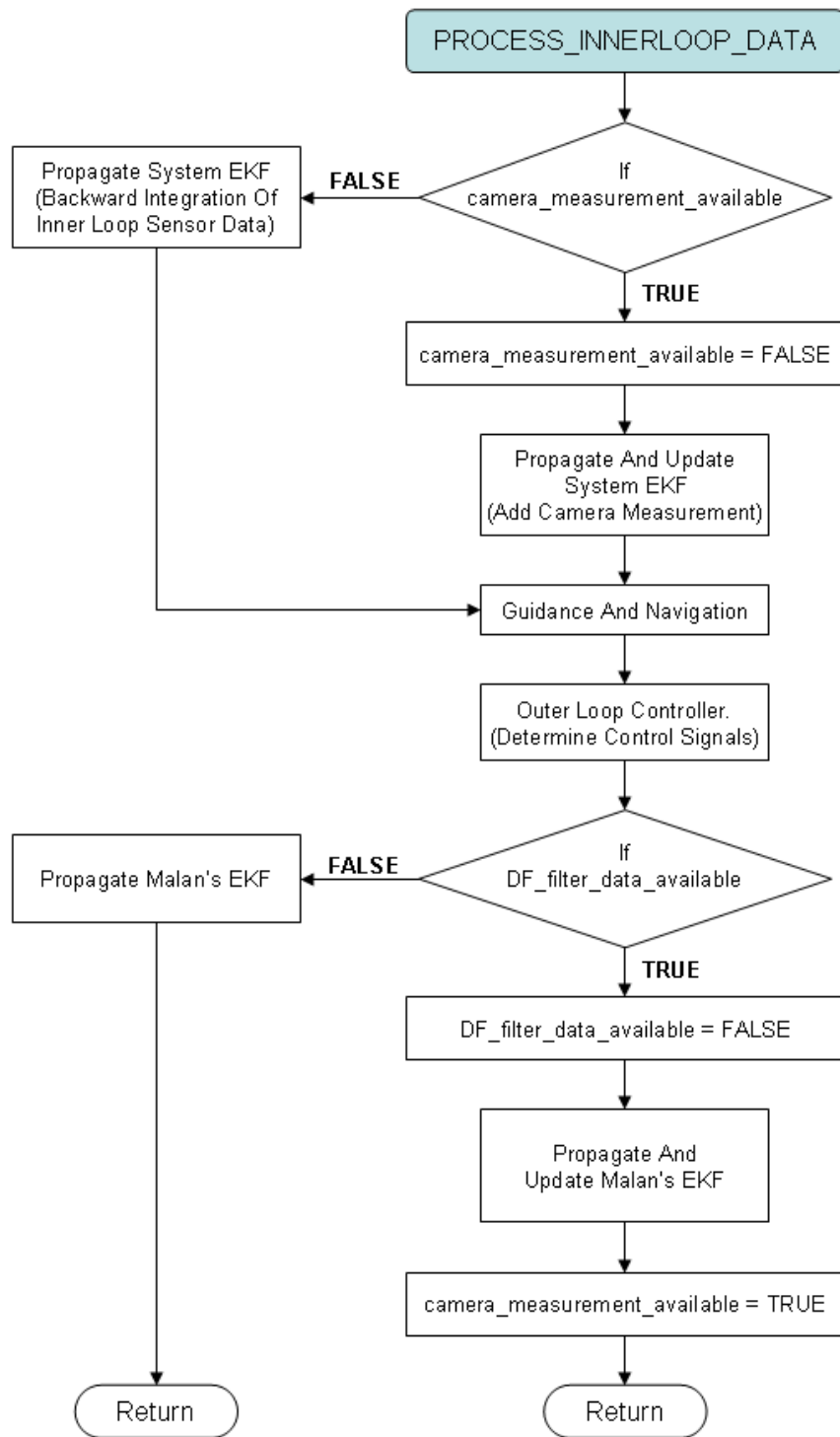


Figure A.3: Process Inner Loop data.

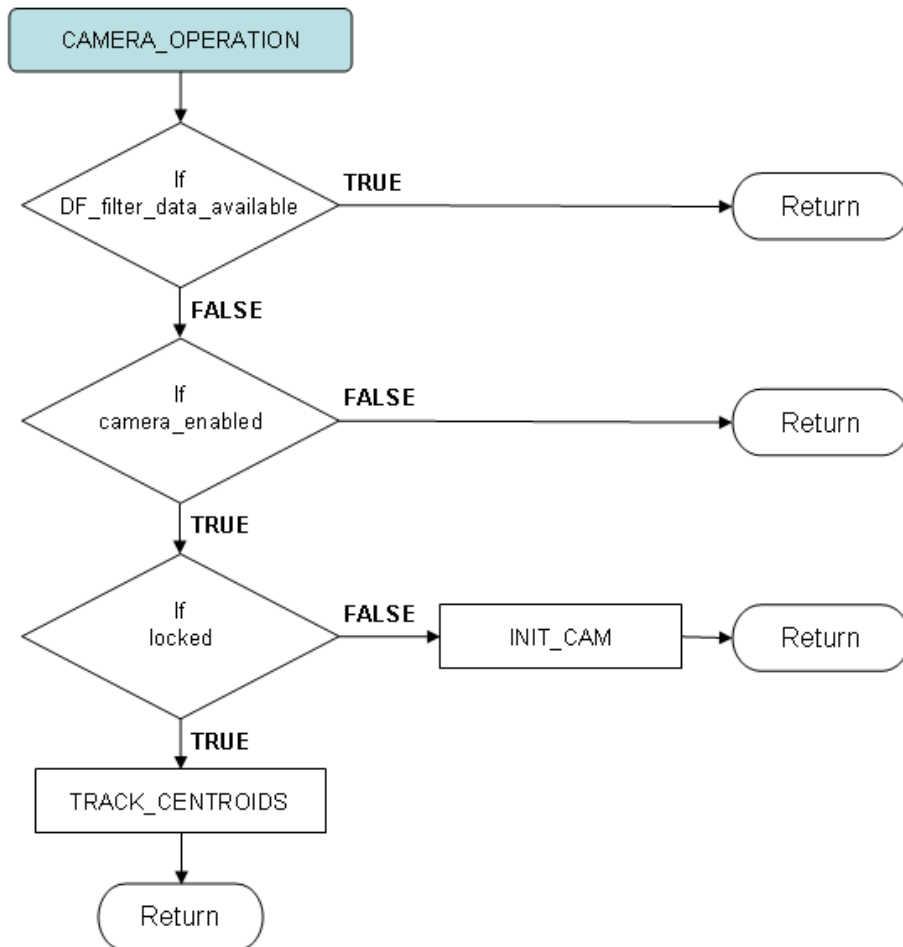


Figure A.4: Camera Operation.

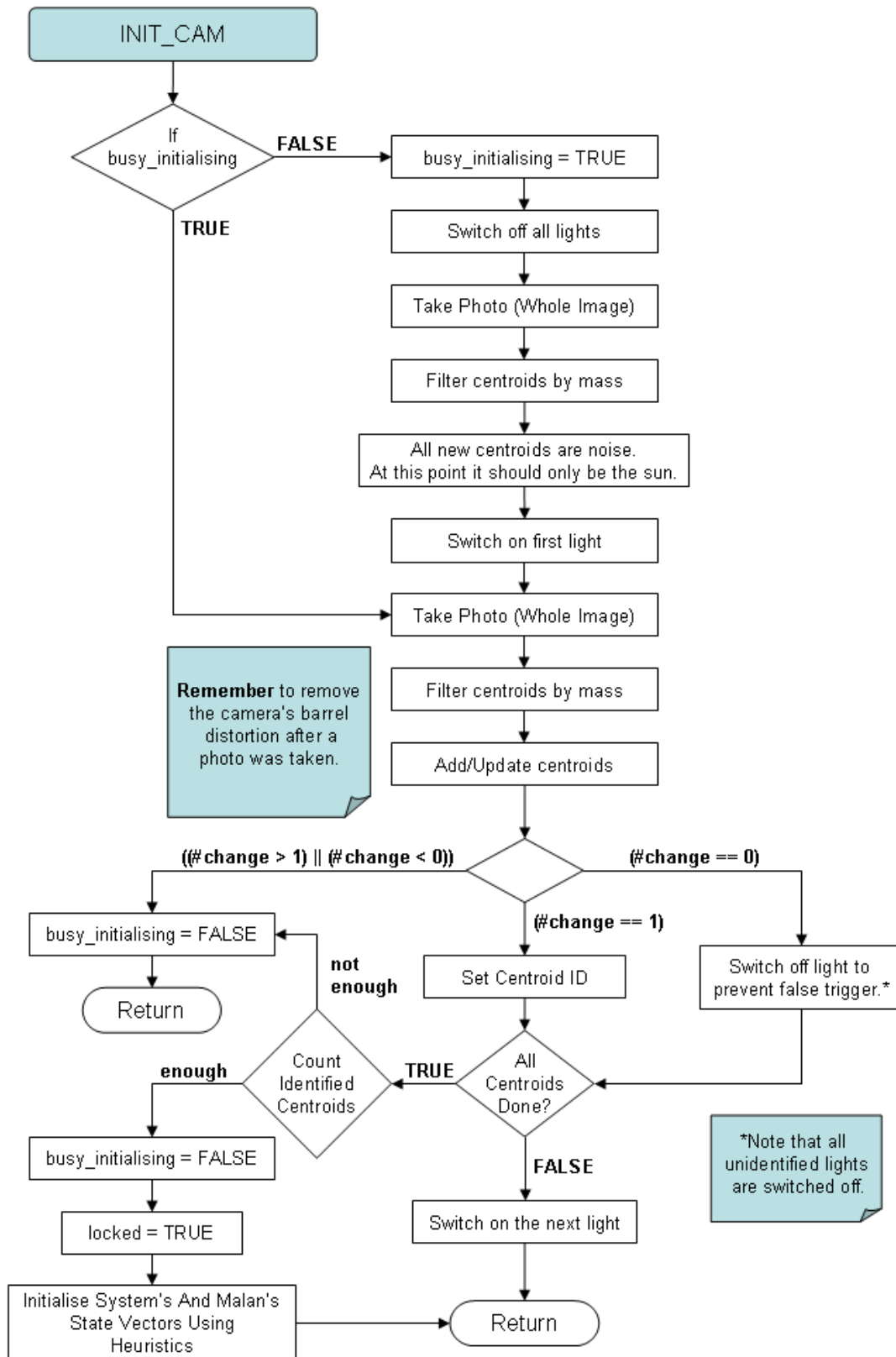


Figure A.5: Initialise Camera.

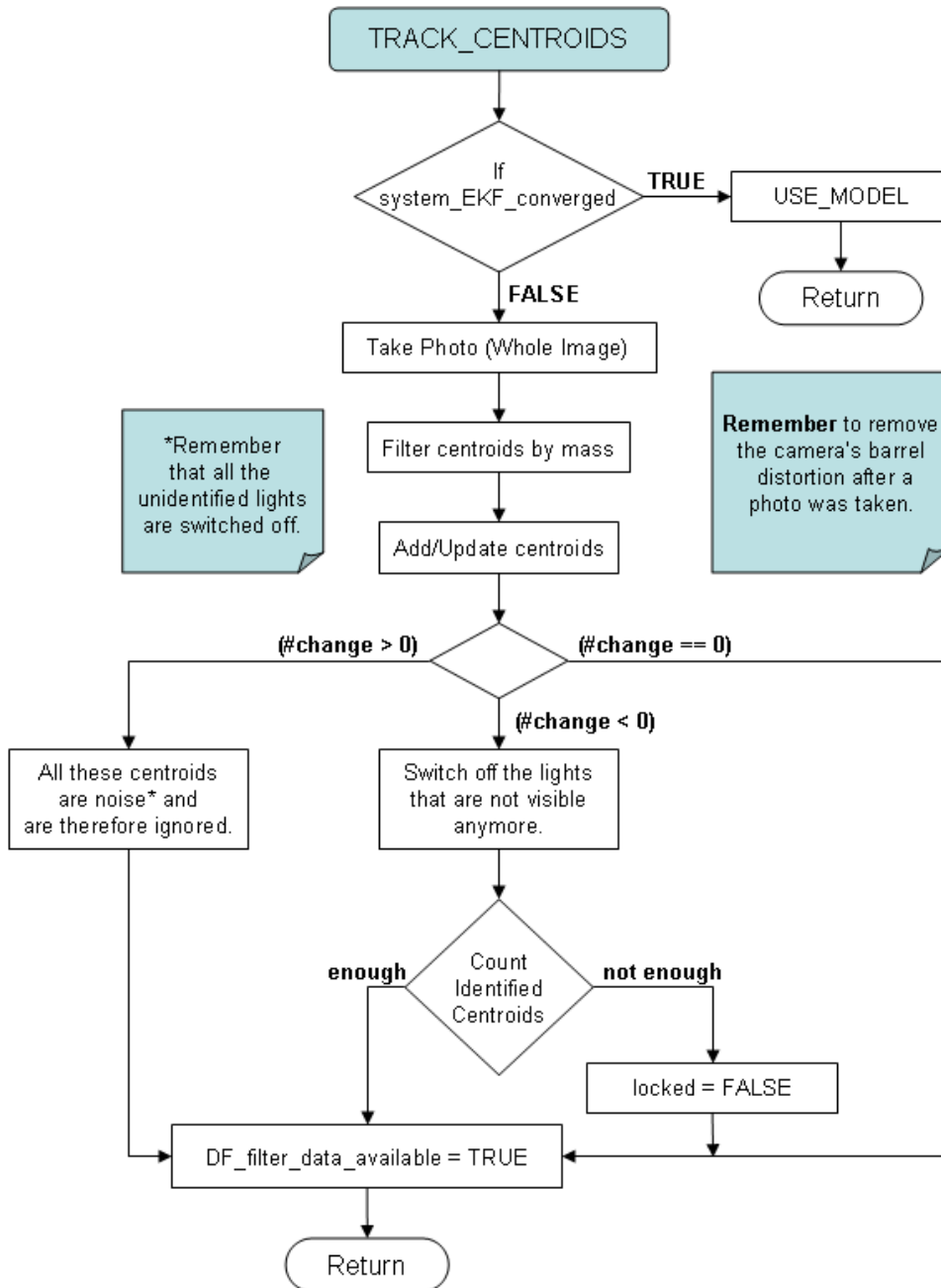


Figure A.6: Track Centroids.

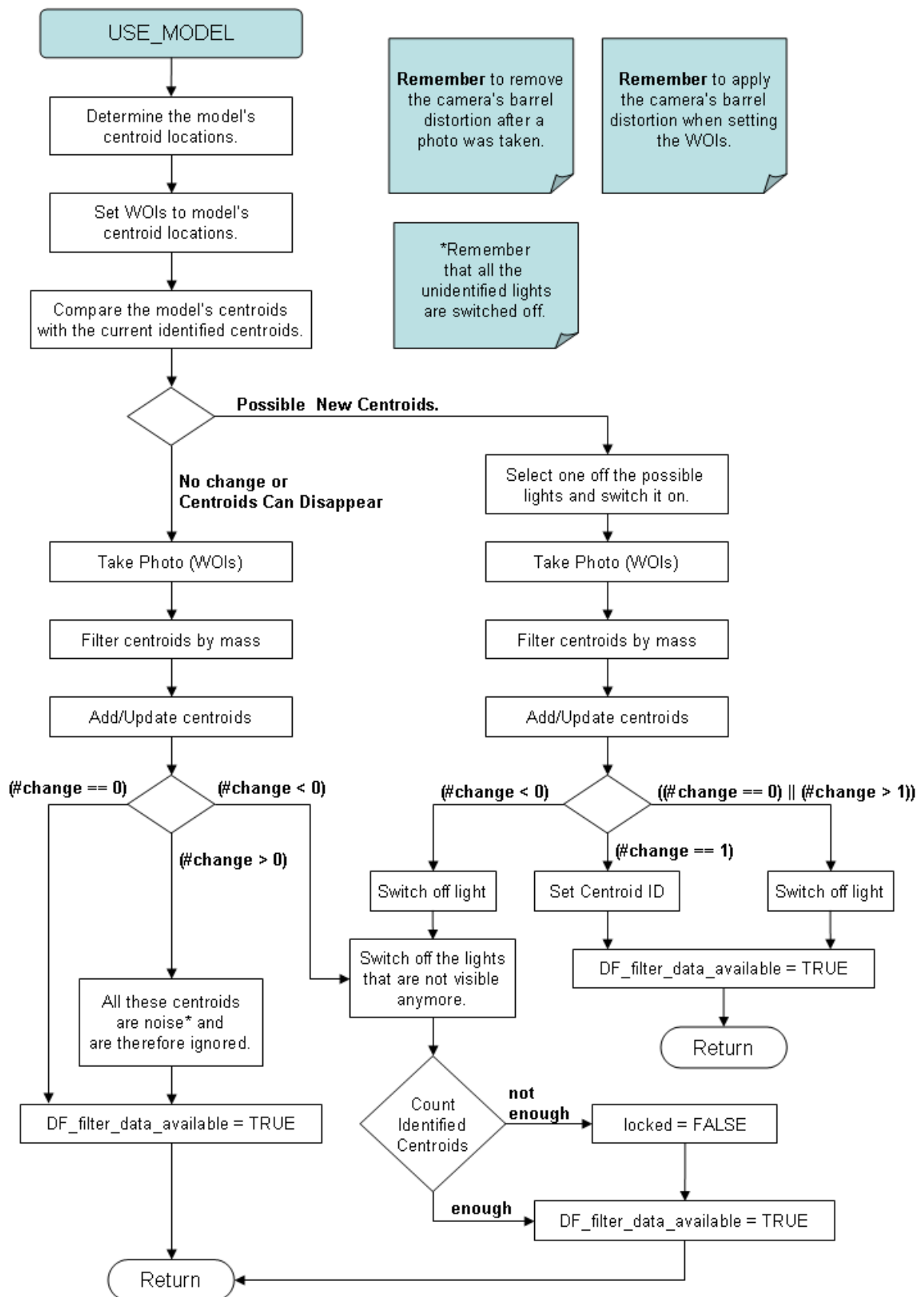


Figure A.7: Use Model.

Appendix B

Derivation of Formulas in Chapter 5

B.1 Movement of stars' centroids due to the rotation and translation of the satellites around one another and the earth

In order to determine the star's displacement in the image, figure B.1 is used.

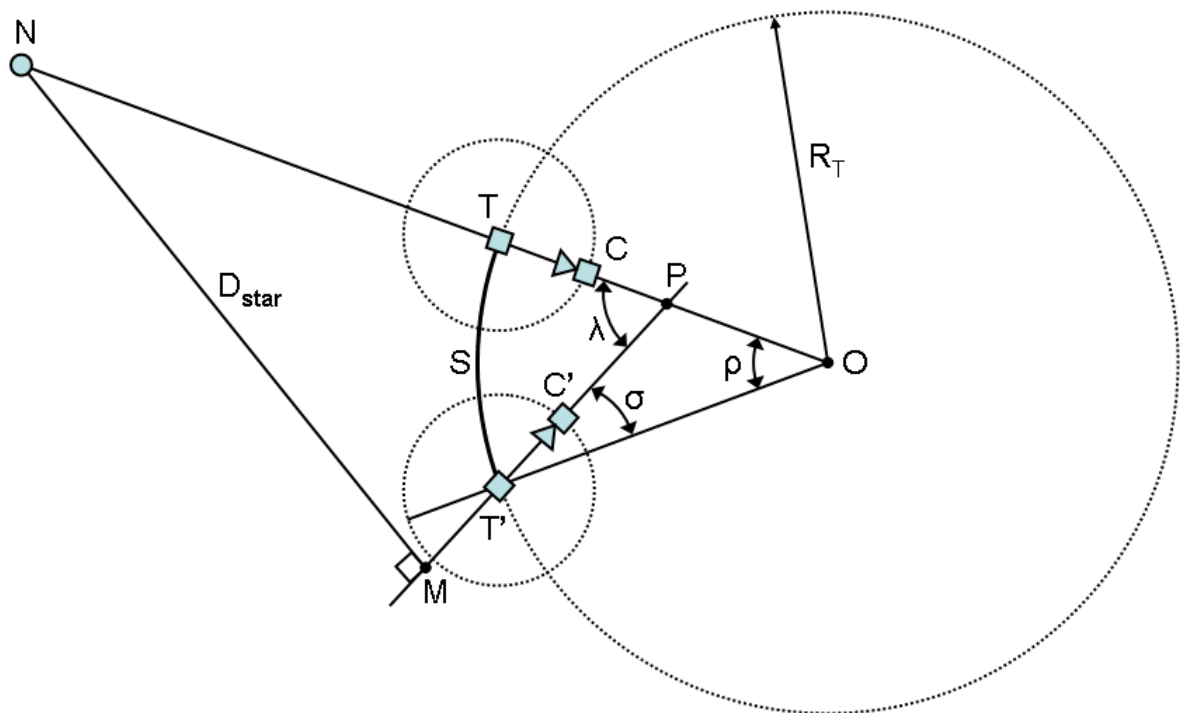


Figure B.1: Movement of a star's centroid due to chaser's and target's rotation around each other and the earth.

Where:

- ΔT is the time elapsed between two successive images.
- The target satellite moves from point T to T' in ΔT , moving through a distance S.
- The chaser satellite moves from point C to C' in ΔT .
- Point O is the earth's centre.

- R_T is the radius of the target satellite's circular orbit.
- ω_{T_E} is the target's angular velocity around the earth.
- D_{star} is the distance, in meters, the the star appears to have moved.
- d_{star} is the displacement of the star in the image.
- The star is located a point N.

Starting with:

$$D_{star} = \overline{NP} \sin(\lambda) \text{ m}$$

The stars displacement in the image can now be determined as follows:

$$\begin{aligned} d_{star} &= \left(\frac{f}{p}\right) \left(\frac{D_{star}}{\overline{MC'}}\right) \\ &= \left(\frac{f}{p}\right) \left(\frac{(\overline{NP}) \sin(\lambda)}{\overline{MC'}}\right) \text{ pixels} \end{aligned} \quad (\text{B.1.1})$$

Relative to the star at infinity:

$$\overline{PC'} \ll \overline{MP}$$

So that:

$$\overline{MC'} \approx \overline{MP} \quad (\text{B.1.2})$$

From B.1.1 and B.1.2:

$$d_{star} = \left(\frac{f}{p}\right) \left(\frac{(\overline{NP}) \sin(\lambda)}{\overline{MP}}\right)$$

And

$$\overline{MP} = (\overline{NP}) \cos(\lambda)$$

Resulting in:

$$\begin{aligned} d_{star} &= \left(\frac{f}{p}\right) \left(\frac{(\overline{NP}) \sin(\lambda)}{(\overline{NP}) \cos(\lambda)}\right) \\ &= \left(\frac{f}{p}\right) \tan(\lambda) \end{aligned} \quad (\text{B.1.3})$$

Now:

$$\rho = \omega_{T_E} \Delta T$$

$$\sigma = \omega_{C_T} \Delta T$$

And:

$$\begin{aligned}\lambda &= \rho + \sigma \\ &= \omega_{T_E} \Delta T + \omega_{C_T} \Delta T\end{aligned}$$

It is expected that:

$$\omega_{T_E} = \omega_{C_T}$$

Therefore:

$$\lambda = 2\omega_{C_T} \Delta T \quad (\text{B.1.4})$$

Substituting B.1.4 into B.1.3 gives

$$d_{star} = \left(\frac{f}{p}\right) \tan(2\omega_{C_T} \Delta T) \text{ pixels}$$

B.2 Movement of stars' centroids due to the rotation of the chaser satellite around its own axis

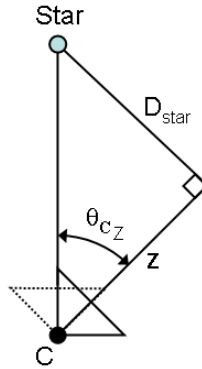


Figure B.2: Movement of star's centroid due to the rotation of the chaser satellite around its own axis.

The movement of a star's centroid in the image due to the rotation of the chaser satellite is determined using figure B.2. To determine the projection of D_{star} onto the the image sensor, start with the basic camera equation:

$$d_{star} = \left(\frac{f}{p}\right) \left(\frac{D_{star}}{Z}\right) \quad (\text{B.2.1})$$

From figure B.2 it can be seen that:

$$D_{star} = Z \tan(\theta_{C_Z}) \quad (\text{B.2.2})$$

Substituting equation B.2.2 into equation B.2.1 results in:

$$d_{star} = \left(\frac{f}{p}\right) \tan(\theta_{Cz}) \text{ pixels}$$

B.3 Movement of the target's lights' centroids due to the relative translation between the satellites

Figure B.3 is used to determine the zoom factor. The zoom factor, Z_F , is defined as the ratio between the distance, d_T , between the new centroids (triangles) and the distance, d_S , between the old centroids (circles), where both d_T and d_S is in pixels:

$$Z_F = \frac{d_T}{d_S} \tag{B.3.1}$$

If the sides of the target is D meters, then from the basic camera equation:

$$d_T = \left(\frac{f}{p}\right) \left(\frac{D}{Z - \Delta Z}\right)$$

$$d_S = \left(\frac{f}{p}\right) \left(\frac{D}{Z}\right)$$

So that:

$$Z_F = \frac{Z}{Z - \Delta Z}$$

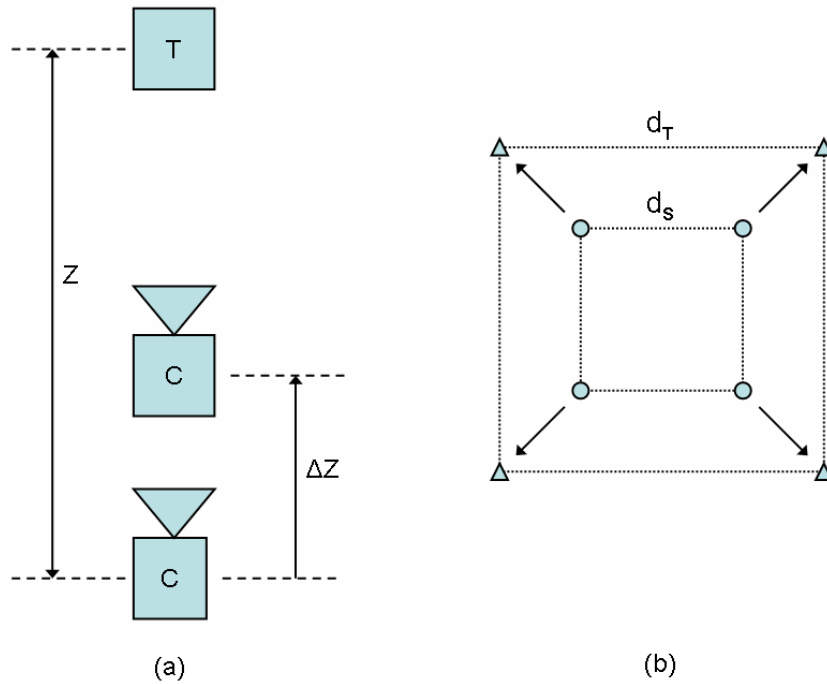


Figure B.3: Movement of the target's lights' centroids due to the relative translation between the satellites.

Appendix C

Rapid Centroid Determination Algorithm

This appendix presents the rapid centroid determination algorithm.

C.1 Some Background

The code given in section C.2 contains the essence of the rapid centroid determination algorithm. Details such as range checking and memory addressing has been omitted for the purpose of clarity. There is also made use of the following 2 functions to encapsulate unnecessary detail:

1. Function *pixel_value(x, y)* returns the intensity value of the pixel in column x and row y .
2. Function *add_and_delete_pixel(x, y)* uses the information of the pixel in column x and row y (such as the pixels position and intensity) to determine the centroid. The centre of intensity is determined by weighing each pixel's position with its intensity. The pixel is then cleared to prevent it from being detected again.

Lastly, the position of the first pixel that passed the threshold is (*startX, startY*).

C.2 The Code

```
searchX = startX;
searchY = startY;

// Add Pixel & Delete
add_and_delete_pixel(searchX,searchY);

// Add pixels to the left & Delete
searchX = startX-1;

while (pixel_value(searchX,searchY) >= THRESHOLD) {
    // Add Pixel & Delete
    add_and_delete_pixel(searchX,searchY);

    searchX = searchX - 1;
}

// Store the left extreme, lu
lu = searchX + 1;

// Add pixels to the right & Delete
searchX = startX+1;

while (pixel_value(searchX,searchY) >= THRESHOLD) {
    // Add Pixel & Delete
    add_and_delete_pixel(searchX,searchY);

    searchX = searchX + 1;
}

// Store the right extreme,ru
ru = searchX - 1;

//First Search Downwards
ld = lu;
rd = ru;

searchMore = TRUE;

// Goto the next row (Downwards)
searchY = startY + 1;
```

```

while (searchMore == TRUE) {
    startX = ld;
    searchX = ld;

    if (pixel_value(searchX,searchY) >= THRESHOLD) {
        //Must search to left & right

        // Add Pixel & Delete
        add_and_delete_pixel(searchX,searchY);

        // Add pixels to the left & Delete
        searchX = searchX-1;

        while (pixel_value(searchX,searchY) >= THRESHOLD) {
            // Add Pixel & Delete
            add_and_delete_pixel(searchX,searchY);

            searchX = searchX - 1;
        }

        // Store the left extreme, ld
        ld = searchX + 1;

        // Add pixels to the right & Delete
        searchX = startX+1;

        while (pixel_value(searchX,searchY) >= THRESHOLD) {
            // Add Pixel & Delete
            add_and_delete_pixel(searchX,searchY);

            searchX = searchX + 1;
        }

        // Store the right extreme, rd
        rd = searchX - 1;
    }
    else {
        // Only search to the right <= rd
        while ((pixel_value(searchX,searchY) < THRESHOLD) && (searchX <= rd)) {
            // Look to the right
            searchX = searchX + 1;
        }
    }
}

```

```

    if (pixel_value(searchX,searchY) >= THRESHOLD) {
        // Store left extreme, ld
        ld = searchX;

        // Add Pixel & Delete
        add_and_delete_pixel(searchX,searchY);

        // Add pixels to the right & Delete
        searchX = searchX+1;

        while (pixel_value(searchX,searchY) >= THRESHOLD) {
            // Add Pixel & Delete
            add_and_delete_pixel(searchX,searchY);

            searchX = searchX + 1;
        }

        // Store the right extreme, rd
        rd = searchX - 1;
    }
    else {
        searchMore = FALSE;
    }
}

// Goto the next row (Downwards)
searchY = searchY + 1;
}

//Now Search Upwards
searchMore = TRUE;

// Goto the previous row (Upwards)
searchY = startY - 1;

while (searchMore == TRUE) {
    if (searchY < 1) {
        break; // break while loop...
    }

    startX = lu;
    searchX = lu;

```

```

if (pixel_value(searchX,searchY) >= THRESHOLD) {
    //Must search to left & right

    // Add Pixel & Delete
    add_and_delete_pixel(searchX,searchY);

    // Add pixels to the left & Delete
    searchX = searchX-1;

    while (pixel_value(searchX,searchY) >= THRESHOLD) {
        // Add Pixel & Delete
        add_and_delete_pixel(searchX,searchY);

        searchX = searchX - 1;
    }

    // Store the left extreme, lu
    lu = searchX + 1;

    // Add pixels to the right & Delete
    searchX = startX+1;

    while (pixel_value(searchX,searchY) >= THRESHOLD) {
        // Add Pixel & Delete
        add_and_delete_pixel(searchX,searchY);

        searchX = searchX + 1;
    }

    // Store the right extreme, ru
    ru = searchX - 1;
}
else {
    // Only search to the right <= ru
    while ((pixel_value(searchX,searchY) < THRESHOLD) && (searchX <= ru)) {
        // Look to the right
        searchX = searchX + 1;
    }

    if (pixel_value(searchX,searchY) >= THRESHOLD) {
        // Store left extreme, lu
        lu = searchX;
    }
}

```

```
// Add Pixel & Delete
add_and_delete_pixel(searchX,searchY);

// Add pixels to the right & Delete
searchX = searchX+1;

while (pixel_value(searchX,searchY) >= THRESHOLD) {
    // Add Pixel & Delete
    add_and_delete_pixel(searchX,searchY);

    searchX = searchX + 1;
}

// Store the right extreme, ru
ru = searchX - 1;
}
else {
    searchMore = FALSE;
}
}

// Goto the previous row (Upwards)
searchY = searchY - 1;
}
```

Appendix D

Photos of the Hardware

D.1 The Chaser and Target

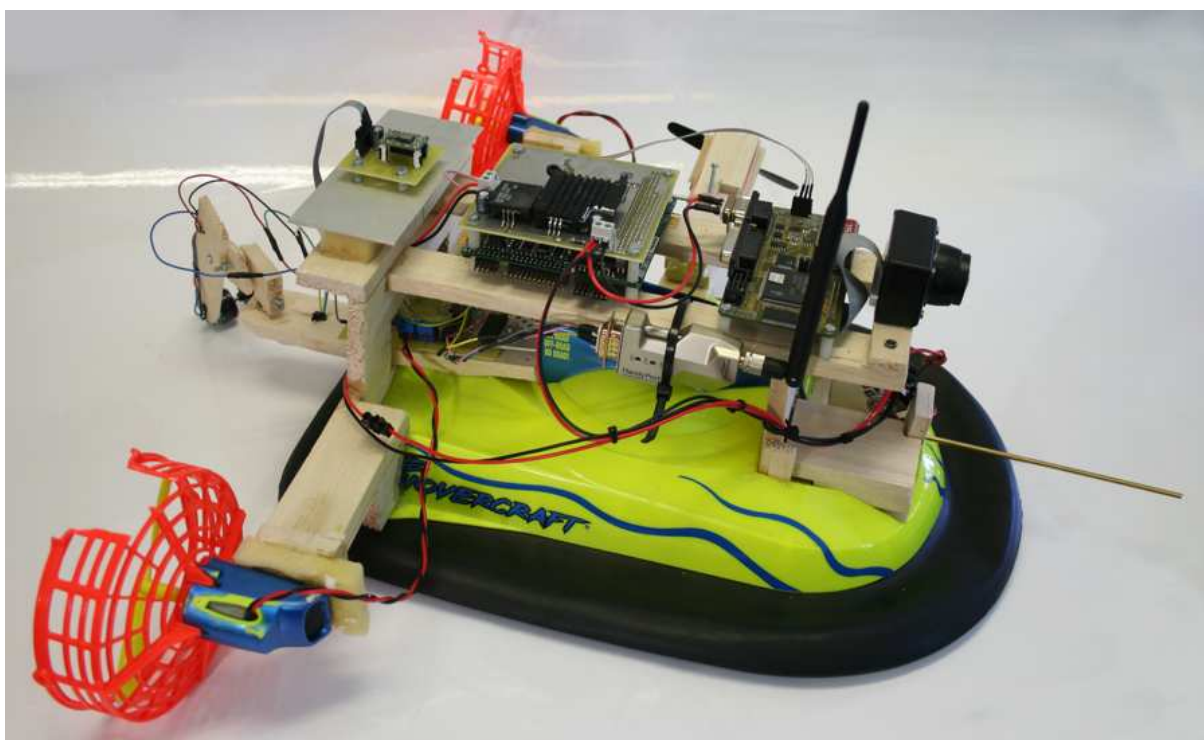


Figure D.1: The chaser.

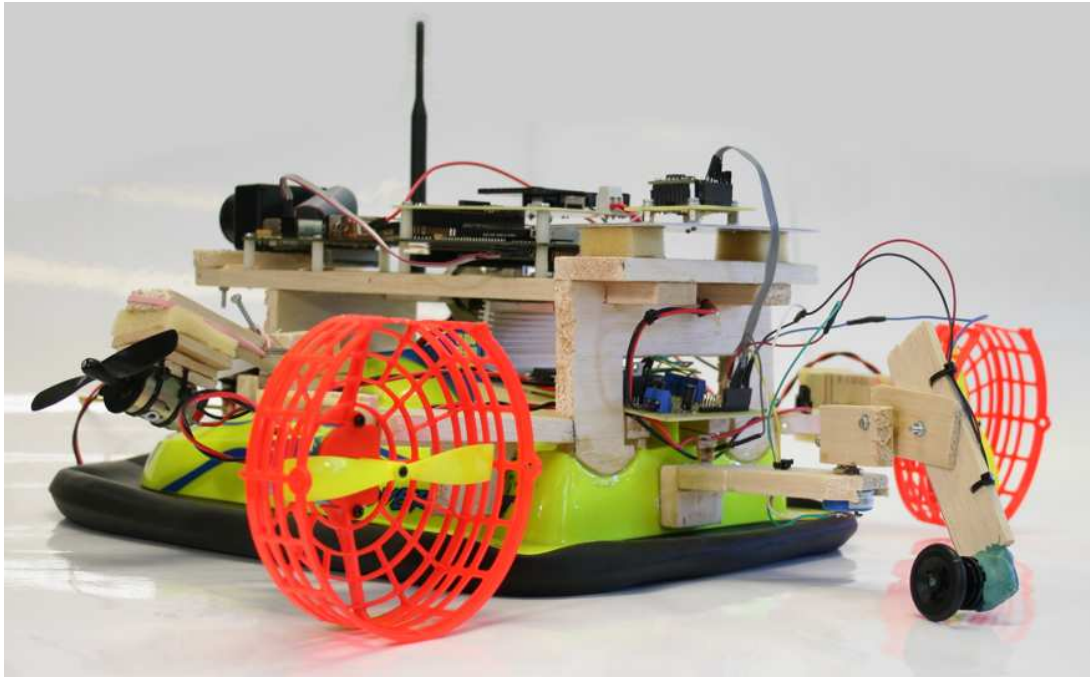


Figure D.2: Another view of the chaser.

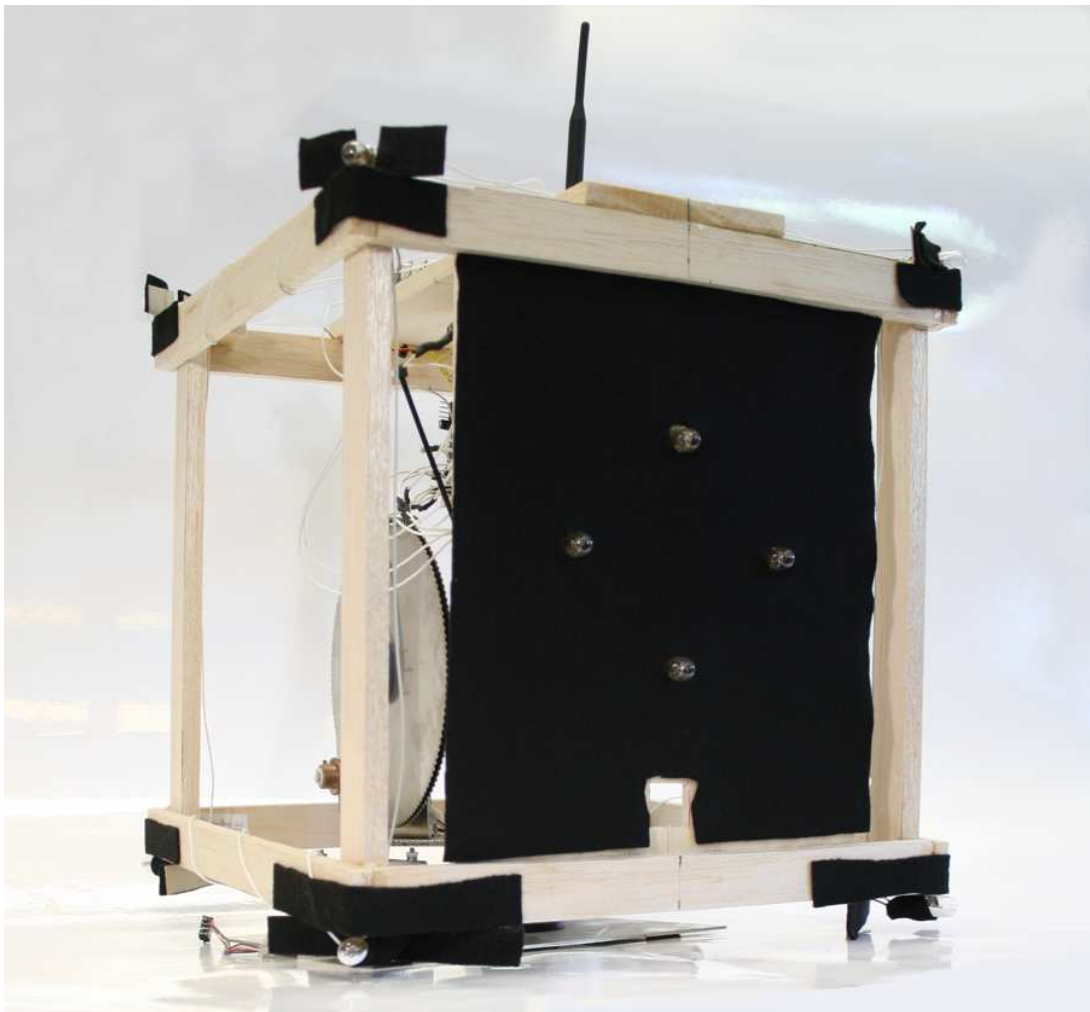


Figure D.3: The target.

D.2 The Docking Mechanisms

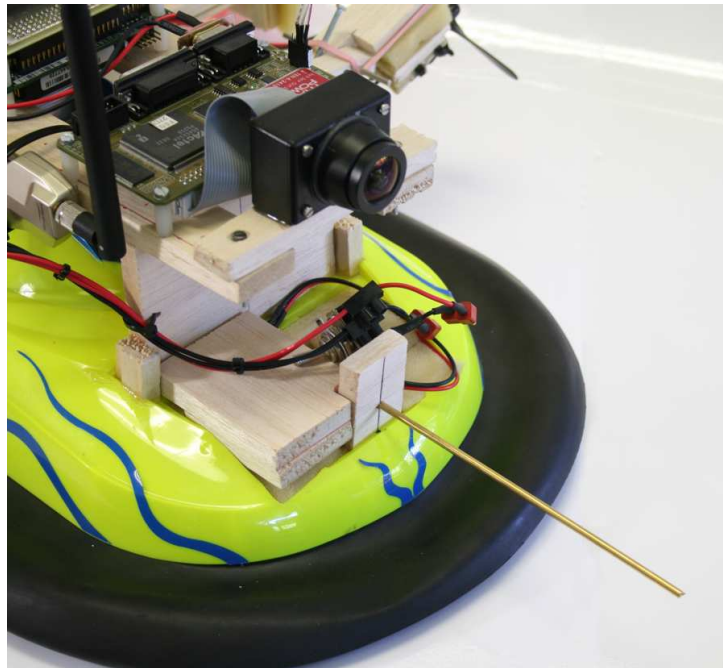


Figure D.4: The pin is the chaser's docking mechanism.

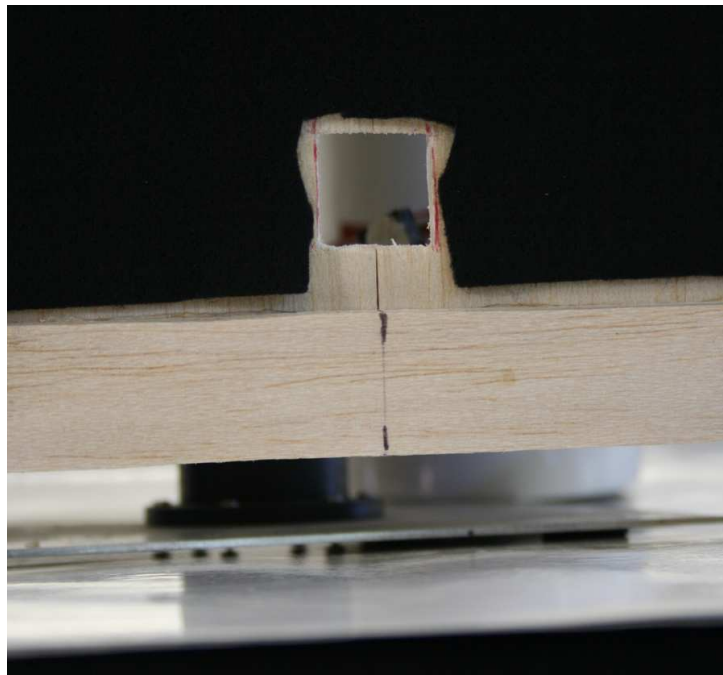


Figure D.5: The hole is the target's docking mechanism.

D.3 The Camera Unit

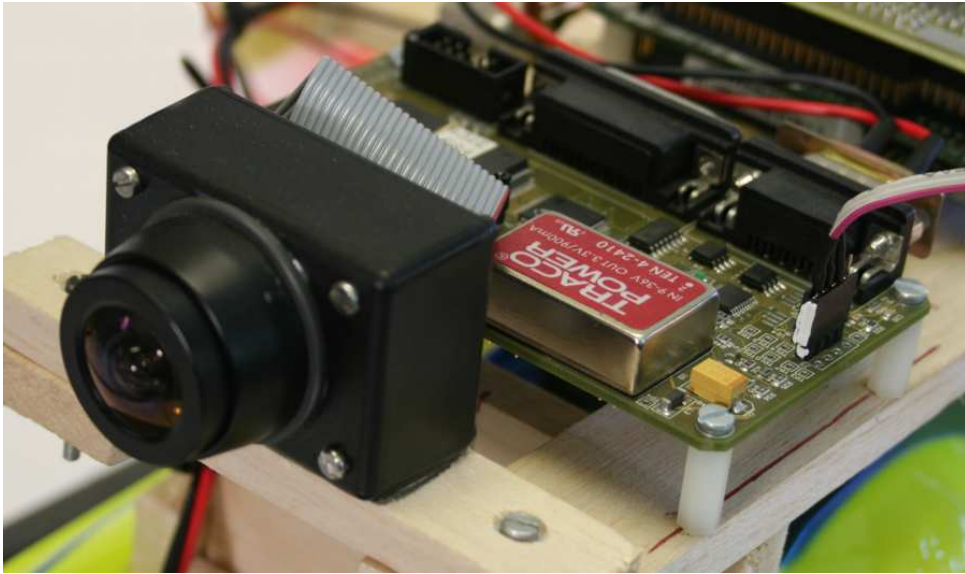


Figure D.6: The camera unit.

D.4 The Velocity Sensor

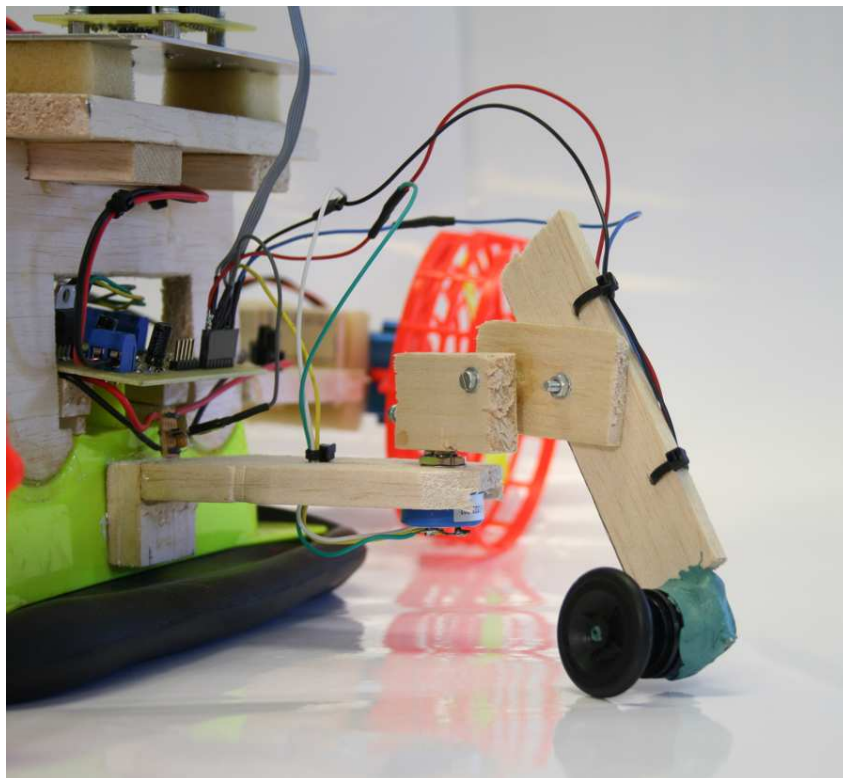


Figure D.7: The velocity sensor.

Appendix E

Homogeneous Vectors and the Projective Camera Matrix

This appendix explains how homogeneous vectors are used and how to use the projective camera matrix, without elaborating on the details of computer vision theory. For more detail refer to [10].

E.1 Basic Knowledge Required

- A 2D point is normally represented by a pair of coordinates e.g. $[x, y]^T$. This is said to be a point in Euclidean 2-space, \mathbb{R}^2

But the same point can be represented by a homogeneous vector in Projective 2-space, \mathbb{P}^2 .

The relationship between the two representations is:

$$\begin{array}{ccc} \mathbb{R}^2 & & \mathbb{P}^2 \\ \left[\begin{array}{c} \left(\frac{x_1}{x_3} \right) \\ \left(\frac{x_2}{x_3} \right) \end{array} \right] & \iff & \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \right] \end{array} \quad (\text{E.1.1})$$

In other words, to convert a point in \mathbb{P}^2 to a point in \mathbb{R}^2 , one must divide the first 2 elements in the homogeneous vector (x_1 and x_2) by the third (x_3) after which the third element is discarded.

- Similarly, a 3D point in Euclidean 3-space, \mathbb{R}^3 , can also be represented by a homogeneous vector in Projective 3-space, \mathbb{P}^3 :

$$\begin{array}{ccc} \mathbb{R}^3 & & \mathbb{P}^3 \\ \left[\begin{array}{c} \left(\frac{x_1}{x_4} \right) \\ \left(\frac{x_2}{x_4} \right) \\ \left(\frac{x_3}{x_4} \right) \end{array} \right] & \iff & \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \right] \end{array} \quad (\text{E.1.2})$$

To convert a point in \mathbb{P}^3 to a point in \mathbb{R}^3 , one must divide the first 3 elements in the homogeneous vector (x_1, x_2 and x_3) by the forth (x_4) after which the forth element is discarded.

E.2 Using the Projective Camera Matrix

Here the use of the projective camera matrix will be shown by means of an example. In this example the projection of a point $[x_p^{CAM}, y_p^{CAM}, z_p^{CAM}]^T$, in the camera axis, on the image sensor will be determined. Starting with:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} \frac{f}{p_x} & 0 & u_0 \\ 0 & \frac{f}{p_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p^{CAM} \\ y_p^{CAM} \\ z_p^{CAM} \end{bmatrix} \quad (\text{E.2.1})$$

Where:

- $[\tilde{x}, \tilde{y}, \tilde{z}]^T \in \mathbb{P}^2$ is the homogeneous vector representing the projection of the point $[x_p^{CAM}, y_p^{CAM}, z_p^{CAM}]^T \in \mathbb{R}^3$ onto the image sensor.
- The camera's focal length is f meters.
- The principle point is given by $[u_0, v_0]^T$ where both u_0 and v_0 are in pixel units.
- The size of each pixel, in meters, is p_x and p_y along the camera's x- and y-axis, respectively.

Multiplication of the matrix with the vector yields:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} \left(\frac{f}{p_x}\right) x_p^{CAM} + u_0 z_p^{CAM} \\ \left(\frac{f}{p_y}\right) y_p^{CAM} + v_0 z_p^{CAM} \\ z_p^{CAM} \end{bmatrix}$$

From the relationship given by E.1.1, the position of point's projection in the image sensor $[x_{sensor}, y_{sensor}]^T \in \mathbb{R}^2$ is:

$$\begin{aligned} x_{sensor} &= \frac{\tilde{x}}{\tilde{z}} \\ &= \frac{\left(\frac{f}{p_x}\right) x_p^{CAM} + u_0 z_p^{CAM}}{z_p^{CAM}} \\ &= \left(\frac{f}{p_x}\right) \left(\frac{x_p^{CAM}}{z_p^{CAM}}\right) + u_0 \text{ pixels} \end{aligned}$$

Similarly:

$$\begin{aligned} y_{sensor} &= \frac{\tilde{y}}{\tilde{z}} \\ &= \left(\frac{f}{p_Y} \right) \left(\frac{y_P^{CAM}}{z_P^{CAM}} \right) + v_0 \text{ pixels} \end{aligned}$$

These are the same results as what was obtained in section 4.3.

E.3 Inserting a DCM and Translation

If the point P is in the observed object's axis, then it first has to be converted to a vector in the camera axis:

$$\begin{bmatrix} x_P^{CAM} \\ y_P^{CAM} \\ z_P^{CAM} \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_P^{OBJ} \\ y_P^{OBJ} \\ z_P^{OBJ} \\ 1 \end{bmatrix} \quad (\text{E.3.1})$$

Where:

- $[x_P^{OBJ}, y_P^{OBJ}, z_P^{OBJ}, 1]^T \in \mathbb{P}^3$ is the point $[x_P^{OBJ}, y_P^{OBJ}, z_P^{OBJ}]^T \in \mathbb{R}^3$ written as a homogeneous vector. $[x_P^{OBJ}, y_P^{OBJ}, z_P^{OBJ}]^T$ is the point P in the object's axis.
- \mathbf{R} is the DCM that converts a vector from the observed object's axis to the camera's axis.
- The translation of the object's axis' origin in the camera axis is \mathbf{t} .

Combining equations E.2.1 and E.3.1 gives:

$$\begin{aligned} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} &= \begin{bmatrix} \frac{f}{p_X} & 0 & u_0 \\ 0 & \frac{f}{p_Y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_P^{OBJ} \\ y_P^{OBJ} \\ z_P^{OBJ} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{f}{p_X} & 0 & u_0 \\ 0 & \frac{f}{p_Y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left(\mathbf{R} \begin{bmatrix} x_P^{OBJ} \\ y_P^{OBJ} \\ z_P^{OBJ} \end{bmatrix} + \mathbf{t} \right) \end{aligned}$$

This is then solved in the same way as shown in section E.2.

Appendix F

Hovercraft Model

In this appendix the hovercraft model is presented that was used to test the inner loop controllers.

F.1 The Simulink Hovercraft Block

A model of the hovercraft was implemented using MATLAB's Simulink. The hovercraft block shown in figure F.1 models the hovercraft's kinematics, the anti-aliasing filters, non-ideal sensor gains, the misalignment between the sensor board's and the hovercraft's axes, the ADCs and the effects of rotations on the accelerometers that are not on the hovercraft's CG.

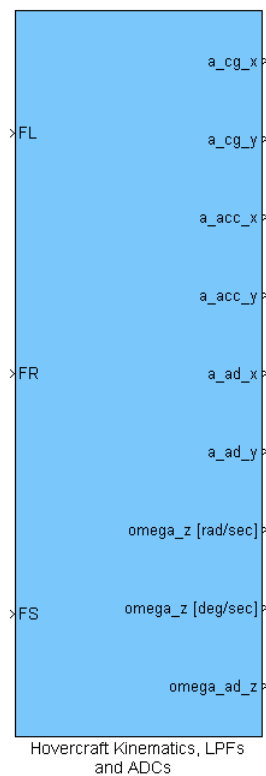


Figure F.1: Hovercraft block.

Referring to figure F.1, here follows a short description of the hovercraft block's inputs and outputs:

- FL, FR and FS are the output force of the left, right and side fan, respectively.
- a_{cg_x} and a_{cg_y} are the acceleration of the hovercraft's CG along its own x- and y-axis, respectively.
- a_{acc_x} and a_{acc_y} are the acceleration experience by the accelerometers, i.e. these accelerations include the effects of angular accelerations and angular velocities around the hovercraft's z-axis.
- a_{acc_x} and a_{acc_y} are the acceleration outputs of the ADCs. (See section F.3 for more detail.)
- ω_{z} [rad/sec] and ω_{z} [deg/sec] are the hovercraft's angular velocity around its z-axis in $\text{rad}\cdot\text{s}^{-1}$ and $^{\circ}\cdot\text{s}^{-1}$, respectively.
- ω_{ad_z} is the angular rate output of the ADC. (See section F.3 for more detail.)

F.2 Using the Hovercraft Block

As was stated in section F.1, the hovercraft block does not include measurement noise or the actuator dynamics. The actuators are modeled using the model from figure 7.8 and measurement noise can be added using MATLAB's White Noise block. This is demonstrated in figure F.2.

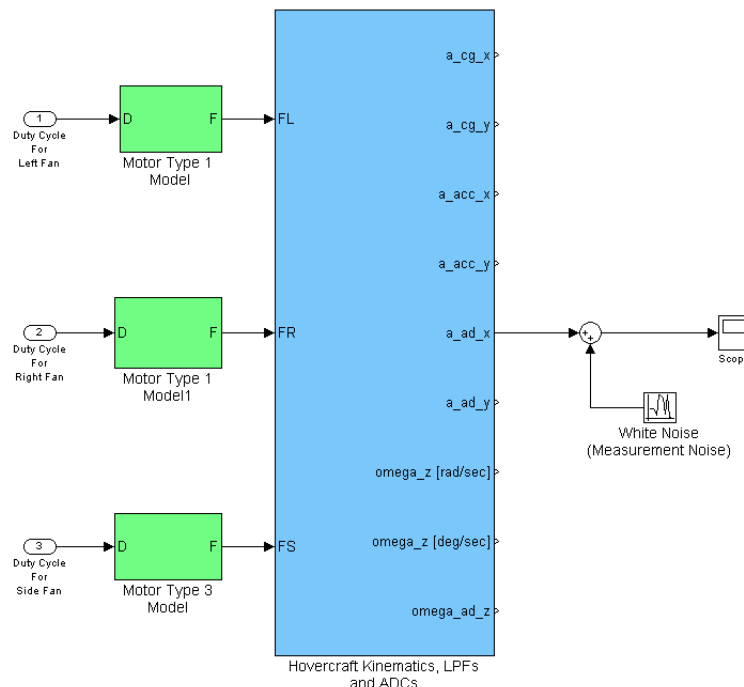


Figure F.2: Using the hovercraft block.

F.3 The Inside of the Hovercraft Block

Figure F.3 shows the inside of the hovercraft block. The following subsections describe the different subsystems.

F.3.1 “Hovercraft Kinematics” block

This block calculates the acceleration of the hovercraft’s CG and the angular acceleration of the hovercraft around its z-axis as result of the 3 forces (F_L , F_R , F_S) acting on the hovercraft:

$$\begin{bmatrix} a_{CGx} \\ a_{CGy} \\ \alpha_z \end{bmatrix} = \begin{bmatrix} \frac{1}{m} & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{m} \\ \frac{r_{CGy}-r_L}{I_{ZZ}} & \frac{r_{CGy}-r_R}{I_{ZZ}} & \frac{r_{CGx}-r_S}{I_{ZZ}} \end{bmatrix} \begin{bmatrix} F_L \\ F_R \\ F_S \end{bmatrix}$$

Where:

- F_L is the output force of the fan on the left hand side of the hovercraft.
- F_R is the output force of the fan on the right hand side of the hovercraft.
- F_S is the output force of the sideways fan on the hovercraft.
- $[0, r_L, 0]^T$ is the position of F_L in the hovercraft’s reference axis.
- $[0, r_R, 0]^T$ is the position of F_R in the hovercraft’s reference axis.
- $[r_S, 0, 0]^T$ is the position of F_S in the hovercraft’s reference axis.
- $[r_{CGx}, r_{CGy}, 0]^T$ is the position of the CG in the hovercraft’s reference axis.
- I_{ZZ} is the hovercraft’s moment of inertia around its z-axis.
- m is the hovercraft’s mass.
- The hovercraft’s CG’s acceleration along its x- and y-axis is a_{CGx} and a_{CGy} , respectively.
- α_z is the angular acceleration of the hovercraft around its z-axis.

F.3.2 “Influence of rotation on accelerometer measurements” block

This block adds the influence of rotations on accelerometer measurements:

$$\begin{aligned} a_{ACCx} &= a_{CGx} - (r_{ACCy} - r_{CGy})\alpha_z - (r_{ACCx} - r_{CGx})\omega_z^2 \\ a_{ACCy} &= a_{CGy} + (r_{ACCx} - r_{CGx})\alpha_z - (r_{ACCy} - r_{CGy})\omega_z^2 \end{aligned} \quad (F.3.1)$$

Where:

- $[r_{ACCx}, r_{ACCy}, 0]^T$ is the position of the accelerometers in the hovercraft’s reference axis.
- The acceleration at the accelerometer’s position is a_{ACCx} and a_{ACCy} along the hovercraft’s x- and y-axis, respectively.
- ω_z is the hovercraft’s angular velocity around its z-axis.

F.3.3 “ADC 1” block

This block simulates the accelerometers’ non-ideal gains, the misalignment between the sensor board’s and hovercraft’s axes, and the ADCs:

$$\begin{bmatrix} a_{ADx} \\ a_{ADy} \end{bmatrix} = G_{acc} \left(\begin{bmatrix} \frac{1}{K_x} & 0 \\ 0 & \frac{1}{K_y} \end{bmatrix} \mathbf{T}_p \begin{bmatrix} a_{ACCx} \\ a_{ACCy} \\ 0 \end{bmatrix} + \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \right)$$

Where:

- a_{ADx} and a_{ADy} are the output of the ADCs.
- K_x and K_y are the accelerometers’ calibration constants.
- \mathbf{T}_p is a partial DCM. \mathbf{T}_p is only the first two rows of $\mathbf{T}_{CHR \rightarrow SB}$, the DCM that gives the sensor board’s orientation with respect to the hovercraft’s axis.
- D_1 and D_2 are the biases of the x- and y-axis accelerometers, respectively.
- G_{acc} is the theoretical gain from the acceleration to ADC’s output.

F.3.4 “ADC 2” block

This block simulates the gyro’s non-ideal gain, the misalignment between the sensor board’s and hovercraft’s axes, and the ADC:

$$\omega_{ADz} = G_{gyro} \left(\frac{t_{33}}{K_\theta} (\omega_{z(deg)}) + D_3 \right)$$

Where:

- ω_{ADz} is the ADC’s output.
- K_θ is the gyro’s calibration constant.
- t_{33} is the element in the 3rd column of the third row of $\mathbf{T}_{CHR \rightarrow SB}$, the DCM that gives the sensor board’s orientation with respect to the hovercraft’s axis.
- $\omega_{z(deg)}$ is the hovercraft’s angular velocity in $^\circ \cdot s^{-1}$.
- D_3 is the gyro’s bias.
- G_{gyro} is the theoretical gain from the angular velocity to ADC’s output.

F.4 Conclusion

This appendix described the hovercraft model used to test the inner loop controllers. The values of all the parameters were determined in chapter 7.

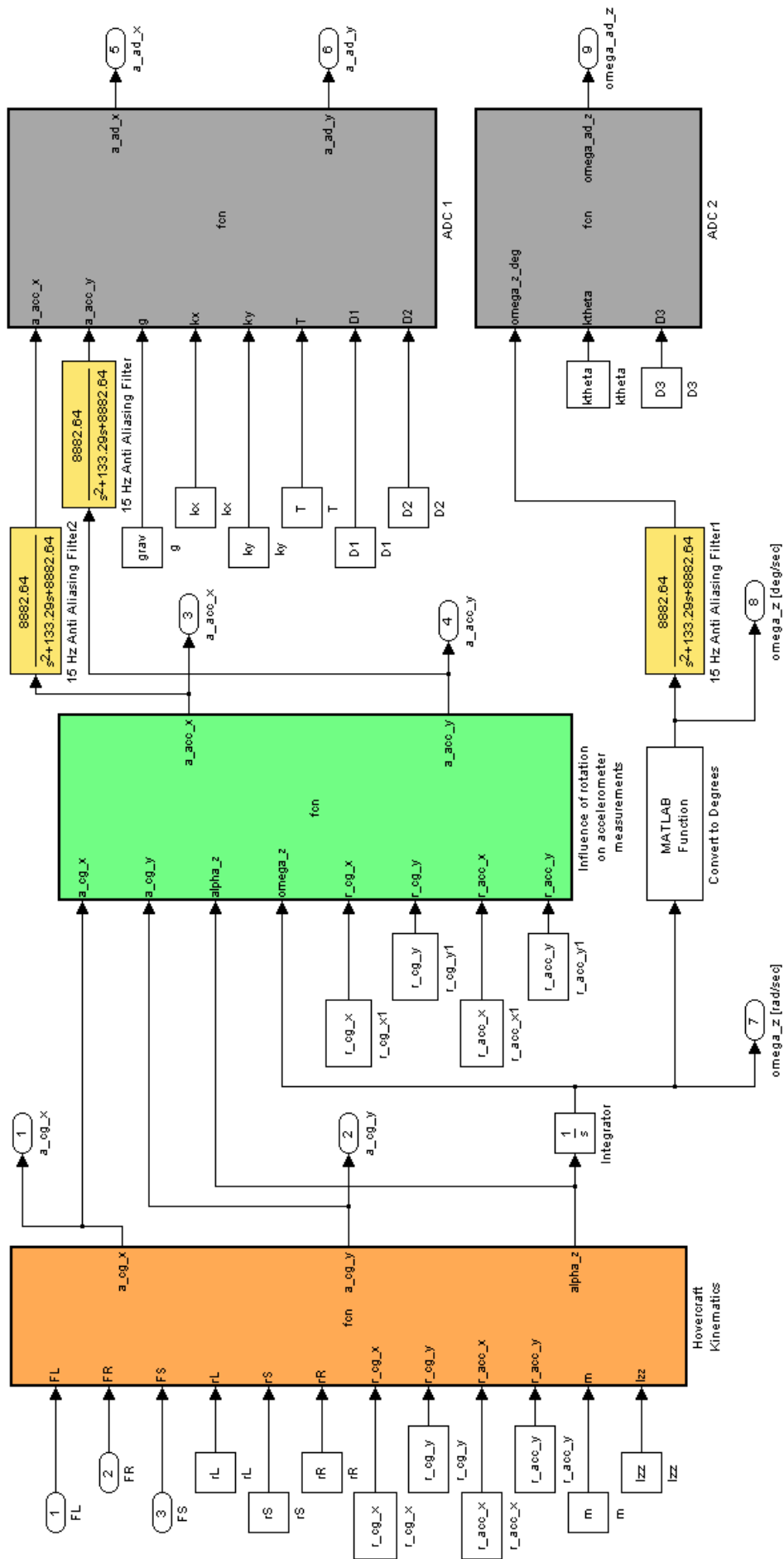


Figure F.3: The inside of the hovercraft block.

Appendix G

Modeling Of Mechanical Vibration

G.1 The Model

The model used to model the mechanical vibrations is shown in figure G.1.

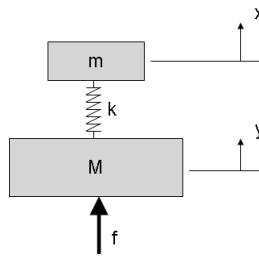


Figure G.1: The vibration model.

Where:

- x is the displacement of the sensor board.
- y is the displacement of the hovercraft body.
- m is the mass of the sensor board.
- M is the mass of the hovercraft body.
- k is the spring constant.
- f is the vibrational force.

This system is described by 2 differential equations:

$$\begin{aligned} M\ddot{y} &= f \\ m\ddot{x} &= k(y - x) \end{aligned}$$

The Laplace transform of the differential equations are given by:

$$Y = \frac{F}{Ms^2} \tag{G.1.1}$$

$$s^2X = \frac{k}{m}(Y - X) \tag{G.1.2}$$

Substituting equation G.1.1 into equation G.1.2 results in:

$$X = \left(\frac{k}{Mm} \right) \left(\frac{F}{s^2 \left(s^2 + \frac{k}{m} \right)} \right) \quad (\text{G.1.3})$$

The next step is to determine the influence of f on \ddot{x} . Define

$$a = \ddot{x}$$

whose Laplace transform is:

$$A = s^2 X \quad (\text{G.1.4})$$

From equations G.1.3 and G.1.4, the transfer function from the mechanical vibration, f , to the acceleration, a , measured by the accelerometer can be determined as:

$$\begin{aligned} H(s) &= \frac{A}{F} \\ &= \left(\frac{k}{Mm} \right) \left(\frac{1}{s^2 + \frac{k}{m}} \right) \end{aligned} \quad (\text{G.1.5})$$

As the vibrations are due to the unbalanced fans, it is expected that the vibrations will have a sinusoidal nature. Using this assumption the measured acceleration due to the vibrations is described by:

$$A = \frac{V \left(\frac{k}{Mm} \right) \omega_F}{\left(s^2 + \frac{k}{m} \right) (s^2 + \omega_F^2)}$$

Where V is the amplitude of the vibration and ω_F is the vibration's frequency.

G.2 Conclusion

From equation G.1.5 it can be seen that the transfer functions gain is directly proportional to the spring constant, k , and indirectly proportional to the mass of the sensor board, m , and the hovercraft, M . Therefore the influence of the vibrations can be reduced by increasing the mass of either the sensor board or the hovercraft or both. Note that it is much more practical to increase the sensor board's mass than the hovercraft's mass as the sensor board only weights a few grams. (It's mass can be increased several fold without influencing the overall mass.)

Because of this, the sensor board has been mounted on a large, thin aluminium plate. This was done so that not only is the sensor board setup's mass increased, but also its moment of inertia around the angular gyros axis.

Also, by increasing the sensor board's mas, m , and decreasing the spring constant, k , the transfer function's cut off frequency is reduced.