Relative Pose Estimation of Spacecraft using ArUco Markers

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Jim Fletcher Jeyakodi David**

Student ID: 915296
Advisor: Prof. Mauro Massari
Academic Year: 2022-23

Contents

Contents **i**

List of Figures **iii**

List of Tables **v**

Abstract **vi**

Abstract in lingua italiana **vii**

1 Introduction **1**

1.1 Relative Navigation in Space 2
1.2 Relative Spacecraft Motion Estimation 5
1.3 Rendering in Space Applications 7
1.4 Thesis Outline 7

2 Mathematical Model **9**

2.1 Reference Frames 9
2.2 Chaser Orbital Dynamics 11
2.2.1 Gravitational Perturbations 11
2.2.2 Atmospheric Drag 12
2.3 Chaser Attitude Equations of Motion 12
2.3.1 Disturbance Torques 12
2.3.2 Sensors 13
2.3.3 Actuators 14
2.3.4 Complimentary Filter 15
2.3.5 Kinematic and Dynamic Model 15
2.4 Relative Spacecraft Motion Model 15
2.4.1 Relative Translational Model 15
2.4.2 Relative Rotational Model 16

3 Observation Model **18**
3.1 Pinhole Camera Model 18
3.1.1 Camera Intrinsic Parameters 20
3.1.2 Camera Extrinsic Parameters 20
3.1.3 Formulation and Solution 22
3.2 ArUco Markers 22

4 Estimation Methodology **25**
4.1 Particle Filter 25

5 Numerical Simulations **27**
5.1 Relative Orbit 29
5.2 Blender Setup 32
5.3 Measurement Model Validation 33
5.4 Filter Dynamical Model Validation 35
5.5 Particle Filter Run 36

6 Conclusions and future developments **39**

Bibliography **40**

A Appendix A **44**
A.1 Calibration Matrix from Blender 44
A.2 Code to interface MatLab output to Blender 44

Acknowledgements **46**

List of Figures

1 PRISMA Spacecraft[5] 4

2 ECI Frame[44] 10
3 LVLH Frame[12] 10
4 Blender and OpenCV Frame[2] 11

5 Pinhole Camera Model[7] 18
6 World and Camera Coordinate frame Relation[19] 21
7 Example of ArUco Marker 23
8 ArUco Relative Coordinate System[34] 24

9 General Algorithm of Particle Filter[20] 25

10 Chaser Orbit 27
11 Angular Velocity of Reaction Wheels 28
12 Angular Velocity of Chaser 28
13 Euler Angles of Chaser 29
14 Relative Position in LVLH Frame 30
15 Relative Velocity in LVLH Frame 30
16 Relative Angular Velocity 31
17 Relative Quaternion 31
18 Blender Setup 32
19 Target Spacecraft with ArUco markers 32
20 Target Spacecraft in Camera perspective 33
21 Generated ArUco Markers 33
22 Detected and Estimated ArUco Marker 34
23 Particle Filter Dynamical Model Validation for Relative Position 35
24 Particle Filter Dynamical Model Validation for Relative Quaternion 36
25 Particle Filter Estimate for Relative Position 36
26 Particle Filter Estimate for Relative Quaternion 37
27 Particle Filter Estimate error for Relative Position 37

28 Particle Filter Estimate error for Relative Quaternion 38

List of Tables

1 Chaser Initial Orbital Elements 27

Abstract

Spacecraft navigation involves using measurements to ascertain the spacecraft's current and future position and attitude. The navigation task is crucial and it involves using sensors, actuators, and dynamical models. In this thesis, the problem of relative navigation between a chaser and target spacecraft which are in close proximity to each other is investigated. A number of applications including formation flying, On-Orbit Servicing (O-OS) of operational satellites, and Active Debris Removal (ADR) takes advantage of models adopted for this problem. In such a scenario, the active satellite which is the chaser would be required to estimate the relative state with high accuracy to meet the controller requirements and to avoid the collision between the spacecrafts.The target spacecraft has fiducial markers called ArUco and the camera on chaser is used to obtain the images of the target. Prior to determining the target/chaser relative position and attitude, images are acquired and analyzed to determine the pose using computer vision algorithms and this pose estimate is then incorporated into the particle filter's measurement equation. Relative position and attitude estimates were obtained with good accuracy using this method.

Keywords: Relative Navigation, ArUco, Particle Filter

Abstract in lingua italiana

La navigazione del veicolo spaziale implica l'utilizzo di misurazioni per accertare la posizione e l'atteggiamento attuali e futuri del veicolo spaziale. Il compito di navigazione è cruciale e comporta l'utilizzo di sensori, attuatori e modelli dinamici. In questa tesi, viene indagato il problema della navigazione relativa tra un inseguitore e un veicolo spaziale bersaglio che sono in stretta vicinanza l'uno all'altro. Numerose applicazioni tra cui il volo in formazione, il servizio in orbita (O-OS) di satelliti operativi e la rimozione attiva dei detriti (ADR) sfruttano i modelli adottati per questo problema. In tale scenario, il satellite attivo che è il cacciatore dovrebbe stimare lo stato relativo con elevata precisione per soddisfare i requisiti del controllore ed evitare la collisione tra i veicoli spaziali. Il veicolo spaziale bersaglio ha marcatori fiduciari chiamati ArUco e la telecamera sul cacciatore viene utilizzato per ottenere le immagini del bersaglio. Prima di determinare la posizione e l'atteggiamento relativi del bersaglio/inseguitore, le immagini vengono acquisite e analizzate per determinare la posa utilizzando algoritmi di visione artificiale e questa stima della posa viene quindi incorporata nell'equazione di misurazione del filtro antiparticolato. Le stime relative di posizione e assetto sono state ottenute con buona precisione utilizzando questo metodo.

Parole chiave: Navigazione relativa, ArUco, Filtro antiparticolato

1 | Introduction

Spacecraft navigation involves using measurements to ascertain the spacecraft's current and future position and attitude. The navigation task is crucial and it involves using sensors, actuators, and dynamical models. Spacecraft absolute navigation refers to the determination of location of spacecraft with respect to an inertial reference frame. Spacecraft relative navigation, or the challenge of determining the relative position and attitude of two separate space objects' reference frames, is the subject of this thesis. Furthermore, big organizations like NASA have recognised the need to greatly increase spacecraft autonomy to support next-generation space missions[13]. In fact, spacecraft missions has to deal with communication delays and no contact with ground stations and autonomy enables to provide less dependency on ground support and more robustness to the mission[31, 41].

In this thesis, the problem of autonomous relative navigation between a chaser and target spacecraft which are in close proximity to each other is investigated. A number of applications including formation flying[45], On-Orbit Servicing (O-OS)[11] of operational satellites, and Active Debris Removal (ADR)[46] takes advantage of models adopted for this problem. In such a scenario, the active satellite which is the chaser would be required to estimate the relative state with high accuracy to meet the controller requirements and to avoid the collision between the spacecrafts. Even though, there are very few missions employing autonomy in space, due to its many benefits, the major space agencies are becoming more interested in a progressive automation of space missions[36].

The target spacecraft can be categorised into cooperative and uncooperative space objects. In case of cooperative target, it can be actively or passively interacting with the chaser spacecraft.The actively cooperative target scenario involves a communication link between the chaser and the target, who both have a vague understanding of their own states. This method can be used in FF and O-OS settings where great precision is required because the navigation performance in this instance is often quite high[29]. In other circumstances, the target may also be collaborating passively by leaving artificial markers on the spacecraft body that the chaser spacecraft can identify and follow[37]. Navigation may be done here as well, and it will be extremely accurate.

The lack of data given by the target spacecraft causes the navigation performance to necessarily suffer when interacting with uncooperative targets. To generate a relative state estimate in this case, complex software techniques must be used with passive or active sensors. However, even a basic understanding of the geometry of the target spacecraft (such a CAD model) can greatly enhance the relative state estimation[36]. The problem of uncooperative targets, which are particularly challenging to approach, is also given special consideration. In reality, the relative navigation problem is extremely challenging to solve because there is no prior information of the target body and there is a lot of ambiguity about its motions.As a result, cutting-edge, customized, technological, and algorithmic solutions must be considered. Prior to determining the target/chaser relative position and attitude, images are acquired and analyzed to determine the pose and this pose estimate is then incorporated into the filter's measurement equation[36].

The need for orbital robotic missions has grown over the past few years for a variety of reasons, including the need to extend the useful life of satellites, reuse special orbital slots, and lower the risk of orbital collision. The satellite's autonomous navigation capacity is a crucial element in such robotic missions since it enables it to carry out relative navigation, maintenance, and repair with little assistance from humans[39]. Relative pose estimation plays a significant role in autonomous GNC for orbiting spacecraft.

1.1. Relative Navigation in Space

This section provides a quick overview of significant previous space missions that used various relative navigation techniques. There have been a lot of technological demonstration missions as a result of the growing interest in autonomous proximity operations. At first, cooperative target missions are examined followed by uncooperative space objects.

The Engineering Test Satellite ETS-7 launched in 1997 is a good example for autonomous relative navigation in space. It was launched by the Japan Aerospace Exploration Agency (JAXA) and was made up of a chaser (Hikoboshi) and a resident space target (Orihime). The coordinated control of the robotic arm and satellite orientation, visual examination, and satellite handling were effectively demonstrated[28]. A 2 meter long, 6 degrees of freedom (DOF) manipulator arm placed on the spacecraft enabled it to conduct a variety of fascinating orbital experiments. It was built and launched by the National Space Development Agency of Japan (NASDA) in November 1997. The mission was divided into two subtasks: performing robotic tests and autonomous rendezvous and docking. Its primary goal was to test robotics technology and show its applicability for unmanned orbital operation and servicing chores. Final stage steering was only accomplished along the V-bar of the Hill frame. Throughout the mission, several abnormalities led to safe mode entry. A planned maneuver moved the chaser spacecraft 2.5 m away from its target due to guidance and navigation failure[28].

In 2005, NASA launched the XSS-11 spacecraft, which was designed to operate in close proximity to its carrier, the second stage of the Minotaur I rocket. To intercept that stage as well as other spent stages near to its orbit, the XSS-11 was outfitted with a camera and a LiDAR[9].

In April 2005, NASA launched the DART spacecraft one week after XSS-11. It developed a goal to show off the technology and software required for Automated Rendezvous and Docking (ARVD) up to a few meters from the target in space. For attitude and translational control, the vehicle used a linear static gain feedback control law, and modulation was accomplished using a pulse width approach. Anomalies and an on-orbit collision were caused by using more fuel than necessary[28]. The mission's initial approach to the target was unsuccessful. This was caused by a skew of almost 0.6 m/sec in the primary Global Positioning System (GPS) receiver's velocity measurement. Its navigation filter diverged as a result, which led to inaccurate thruster firing. DART eventually impacted the target due to ineffective collision avoidance. This shows categorically that ARVD is not yet developed enough to be a secure technology. It also emphasizes how important it is to create approach trajectories that assure collision avoidance for some typical failures while also lowering the possibility of catastrophic failures[28].

The DARPA Orbital Express Advanced Technology Demonstration project is an important mission that is focused on ARVD and the use of robotic technology in space after DART. The mission, which was launched in March 2007, was intended to support a variety of future commercial space initiatives and security needs for the United States. This mission evaluated the viability of utilizing robotic, autonomous on-orbit refueling and satellite reconfiguration. Critical technologies for OOS systems, such as servicing interfaces, autonomous GNC systems, close proximity operations were shown by the Orbital Express mission[28]. Multiple docking scenarios of increasing difficulty were attempted in order to test the system's capacity for autonomous fault reaction. It has shown that it is capable of approaching, relocating, autonomously capturing the target satellite, and using robotics for subsequent servicing. This project encompasses important technologies for robotic OOS of co-operatively built assets in the future, which are still considered non-cooperative targets because they cannot be manoeuvred to speed up the berthing or docking procedure. The incorporation of two distinct patterns on the target spacecraft to estimate its pose is particularly noteworthy: one larger size pattern was used during far approaches, and the other was specifically created to support visual navigation during proximity or close range operations when the chaser is within a range of 5 meters from the target[28]. A DARPA project called the SUMO is intended to do OOS with various client spacecraft without the need for servicing aids.

PRISMA, which was launched in 2010, is another significant mission exhibiting autonomous spacecraft proximity operations. In order to demonstrate the viability of using relative navigation techniques (such as GPS-based, RF, or vision-based to enable future FF operations), the chaser made a variety of maneuvers and relative encounters around the target spacecraft throughout the course of the mission[15, 36].

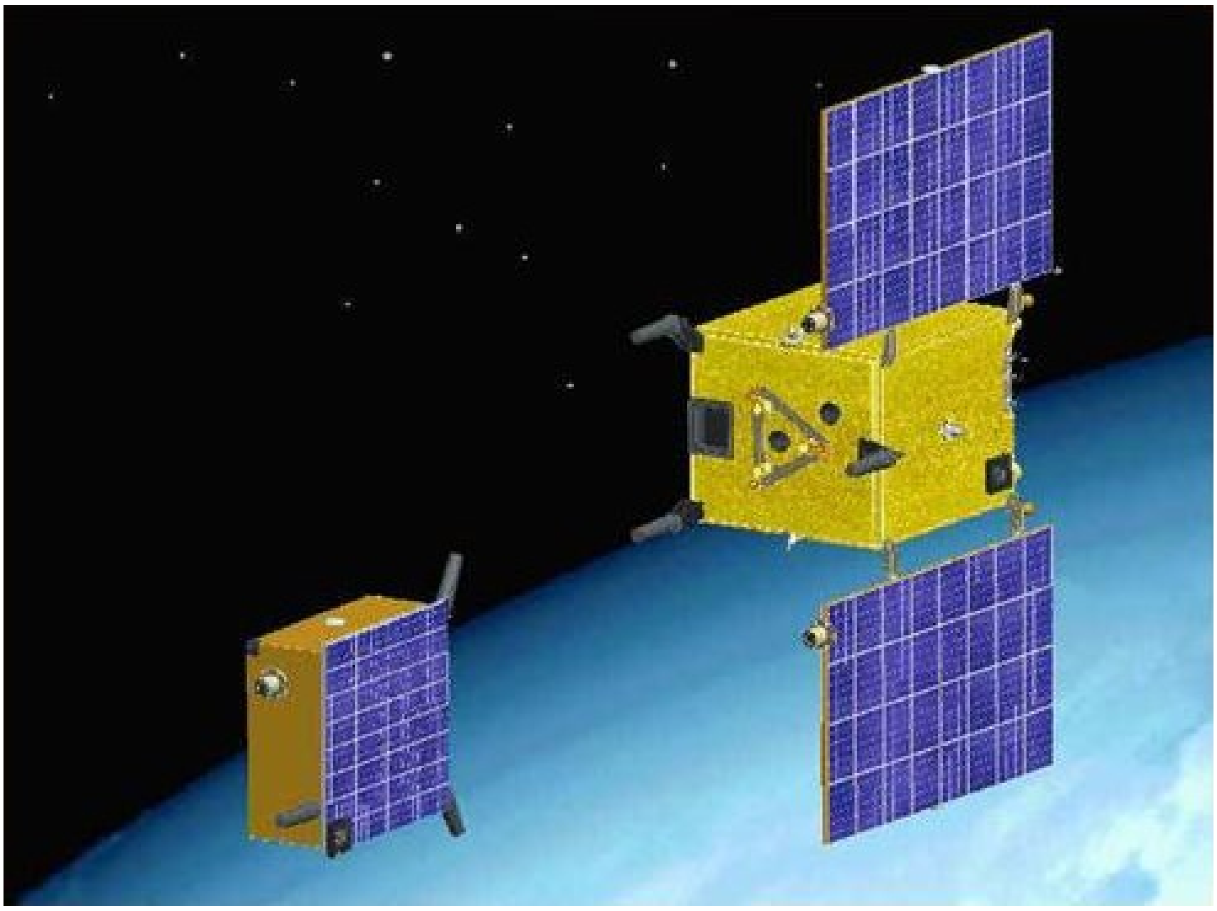


Figure 1: PRISMA Spacecraft[5]

Additionally, docking with the International Space Station (ISS) has grown in importance as a challenge to be solved, and in 2009, the Canadian Space Agency and NASA proposed a LiDAR-based solution. TriDAR (Triangulation and Light Detection and Ranging Automated Rendezvous and Docking) is a combined system that consists of a thermal imager and a LiDAR for automated rendezvous and docking[38]. To recover crucial information like the relative range and location between the spacecraft and the ISS, up to 1 km distance, the shape of the target is presumed to be known and is compared to the estimated one. This technology was used by the Cygnus spacecraft on three Space Shuttle missions[10]. A comparable sensor component called DragonEye is also used by the Dragon spacecraft. Up to 750 m, relative GPS navigation is employed; after that, LiDAR, DragonEye, and thermal cameras are used to follow the ISS[16].

1.2. Relative Spacecraft Motion Estimation

Relative pose estimation for the spacecraft is used in problems of formation flying, docking and close proximity operations and is becoming more important for such problems. If the target is known and cooperative, the problem of estimating is much easier compared to unknown and uncooperative space objects. In addition, this type of estimation requires relative sensors such as light beacons, differential GPS receivers or artificial markers on the target spacecraft[35]. In this thesis, we look at the problem that the target is known and cooperative and that there are artificial markers on the target spacecraft which is viewed by the camera on the chaser spacecraft. There is a lot of literature dealing with the cooperative objects and it will be discussed briefly in the following section.

The rationale behind cooperative pose determination methods is that a small subset of features that are retrieved from the acquired datasets can be used to estimate the relative attitude and position of a target spacecraft body frame with respect to a chaser spacecraft sensor reference frame. They specifically correspond to passive/active artificial markers that are installed on the target surface and whose 3D position in target body frame is known; these markers often produce more distinct echoes than natural features (such as corners)[33]. As a result, the pose determination procedure must incorporate methods for appropriately matching the observed markers within a catalog that is maintained on board.

The line of sight of the markers' centroid in sensor frame is first extracted from monocular pictures using processing techniques. This can be accomplished by employing fairly common image processing methods, which typically involve steps for feature detection and image segmentation. Of course, the geometry and configuration of the markers will determine which strategy is best. The real markers are then matched to the extracted features (by utilizing knowledge of either their shared geometry or individual shape), resulting in a predetermined number (n) of 2D to 3D point correspondences[33]. The process of determining the relative position and orientation of a camera with respect to the observed scene (for example, the target), also known as extrinsic camera calibration[42], can be dealt with by solving the Perspective-n-Point (PnP) problem, provided that there are n matches between real world and image points. Fischler and Bolles first proposed this task to ascertain the 3D position of the image points in the sensor frame if their corresponding 3D position in a real-world coordinate system (for example, the target frame) is already known[22]. Recently, however, this task has taken on a more broad meaning within the involved scientific community, i.e., it is equivalent to pose determination.

Strong attempts have been made recently to identify PnP-solutions that are concurrently precise, quick, and robust to outliers.[21, 25, 30] This is crucial in order to eliminate incorrect matches brought on by inaccurate marker detections. This might happen in the space environment because of the strange visibility circumstances.

Monocular and 3D techniques both involve establishing the relative attitude and position between two reference frames given a set of n 3D-to-3D point correspondences, so once the markers' 3D positions in sensor frame are known, there are no conceptual distinctions between the two[33]. It goes without saying that in order to extract the cooperative markers and perform the matching function, 3D techniques still need to process the collected 3D data in advance.

If n is three, in the sensor and target frames, at least two directions (denoted by the relative position vectors of the marker) can be computed[33]. By using TRIAD[40], a deterministic method with numerous variations, this information can be used to recreate the relative rotation matrix. Potentially higher unique unit vectors can be computed in both reference frames if n is more than three. The QUEST algorithm[40], which has also undergone various improvements, is one probabilistic strategy that can be used to take use of these multiple vector measurements for relative attitude determination. Given the relative rotation matrix, it is possible to characterize a marker's location in terms of the target and sensor origins in the same reference frame, with the result that the difference between the two represents an approximation of the relative translation. The pose estimate method can be carried out directly utilizing the 3D position information of the markers, it is important to note. In particular, the best transformation required to line up two patterns of 3D points can be used to calculate the 6-DOF pose parameters[33]. This can be accomplished by leveraging techniques like the Singular Value Decomposition to minimize the cost function, which is commonly expressed as the sum of Euclidean distances between corresponding points[14, 27, 43]. It is important to note that these methods, which are widely used for a variety of robotic applications, are also appropriate for non-cooperative pose determination tasks, for which point correspondences are less accurate and the size of the 3D point pattern is much larger than for cooperative tasks. However, in a more general scenario, the PnP can be viewed as a black box that, given the Markers' 3D position in target frame and their line of sight in sensor frame, outputs the relative position vector and rotation matrix directly[33].

1.3. Rendering in Space Applications

A quick overview of rendering softwares is given in this final part. Accurate space imaging software is a crucial tool when working with visual relative navigation techniques; in fact, because of its quick prototyping capabilities, it can be a beneficial instrument to deploy in conjunction with an experimental GNC facility. It is also frequently highly efficient to run rendering engines on such GPUs, which, when clustered in a server, may give a significant amount of computational capacity, thanks to the ongoing advancements in graphic processor technology[38].

There are currently some space imaging programs available, such as PANGU[6], a program developed by the University of Dundee in Scotland with assistance from the ESA. To test vision-guided navigation, guidance, and landing systems, it may produce camera and LiDAR images of various planetary bodies and spacecraft. Only ESA projects are permitted to utilize the software, which is licensed and cost-free.

Airbus Defence and Space has created additional software that is comparable. Its name is SurRender[8], and it makes use of a ray-tracing engine to produce pictures specifically for space scenes including planetary approach, landing, and in-orbit rendezvous. On request, the software is available for free usage. Both programs quantitatively include different effects brought on by the environment in which the camera is situated. Examples include internal light scattering and lens distortion in cameras, motion blur, electronic sounds, and rolling shutter[38].

In this thesis, synthetic images of target spacecraft are created using Blender. In the entertainment sector, Blender[3], a free and open source 3D creation suite, is widely used for animation, film production, and 3D modeling. Even though Blender is a widely used reference tool for 3D modeling of satellites and asteroids, to the author's knowledge no significant space-related tools have been built in it.

1.4. Thesis Outline

The aim of this thesis is the relative pose estimation of the target spacecraft with respect to chaser spacecraft using the ArUco markers on the target spacecraft body. A particle filter is designed to improve the estimated pose given the computer vision algorithms. In Chapter 2, the reference frames used in the thesis is presented and the equations of motion for the chaser spacecraft and the relative dynamical models are described. The various environment disturbances considered along with the models is also discussed. In Chapter 3, the observational model is presented and a detailed description of ArUco markers along with the generation, detection and estimation of those markers is illustrated. Chapter 4 presents the filtering method employed in this thesis and the general theory with the algorithm and the tool it is implemented in is described. The results are presented in detail in Chapter 5 of this report.

2 | Mathematical Model

In order to determine the relative motion of spacecrafts, accurate models are required. In this thesis, the target and chaser are assumed to be in circular orbits and in close proximity. Then the most famous relative dynamics model by Clohessy and Wiltshire[17] is used. In case of chaser spacecraft dynamics, the environmental disturbances are considered and Cowell's method is used to integrate the equations of motion. Runge Kutta-4 method is used to propagate the states.

2.1. Reference Frames

Standard reference frames are used in this thesis. Earth Centered Inertial frame is used to denote the chaser position and the propagation of chaser states (translation and rotation) is done in this frame. There are two spacecrafts considered: chaser and target. Chaser spacecraft is the active one and target spacecraft is the passive one. The body frames of both spacecrafts are considered to be Local Vertical - Local Horizontal (LVLH) frame. The relative propagation is done in LVLH frame and it is input into Blender software. It has a camera which has it's own sensor frame and the openCV functions used in MatLab has a different camera frame and those frames are also presented here for completeness.

- Earth Centered Inertial Reference frame (ECI): $\hat{I}$ directed from the center of the Earth towards the vernal equinox, $\hat{K}$ is normal to the equatorial plane and towards the north pole and $\hat{J}$ completes the right hand triad.

- Local Vertical - Local Horizontal frame (LVLH): The origin is the center of mass of the spacecraft, $\hat{x}$ directed from the origin radially outward, $\hat{z}$ is normal to orbital plane positive in the direction of angular momentum vector and $\hat{y}$ completes the right hand triad. For a circular orbit, $\hat{y}$ aligns along the velocity vector.

-Blender camera frame: The camera looks along the -Z axis, Y oriented up and X completes the frame.

-OpenCV camera frame: Camera principal axis aligned with positive Z axis , Y axis is oriented downward and X axis completes the right-handed frame.

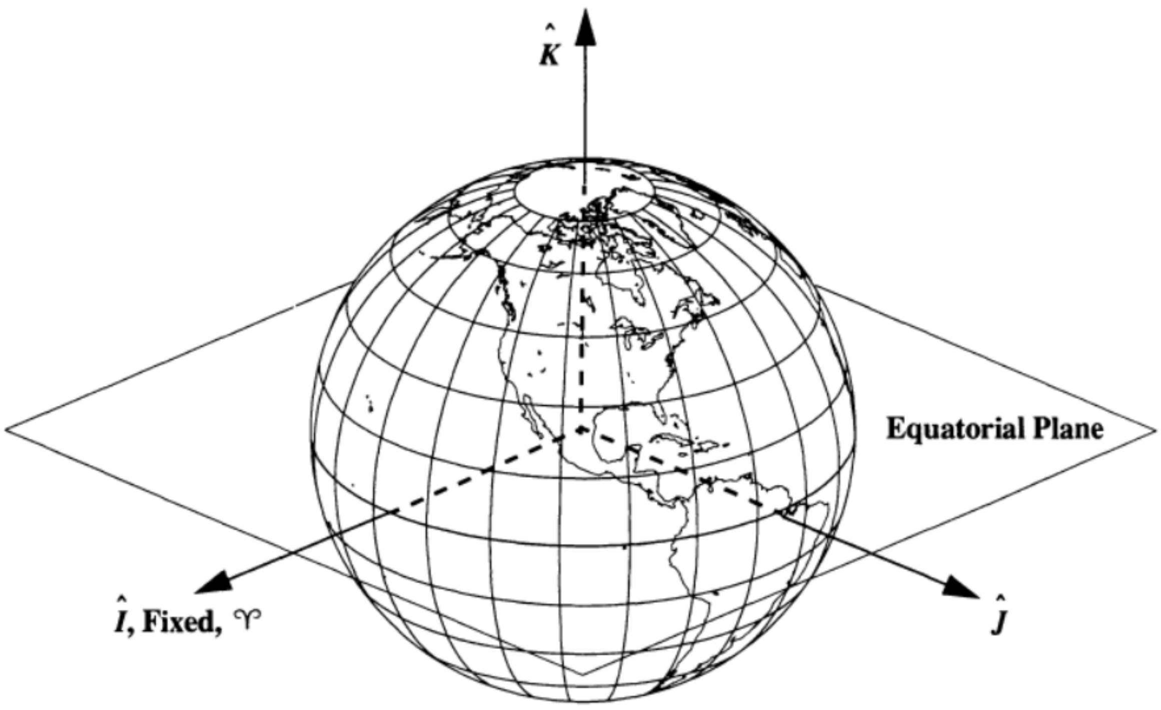


Figure 2: ECI Frame[44]

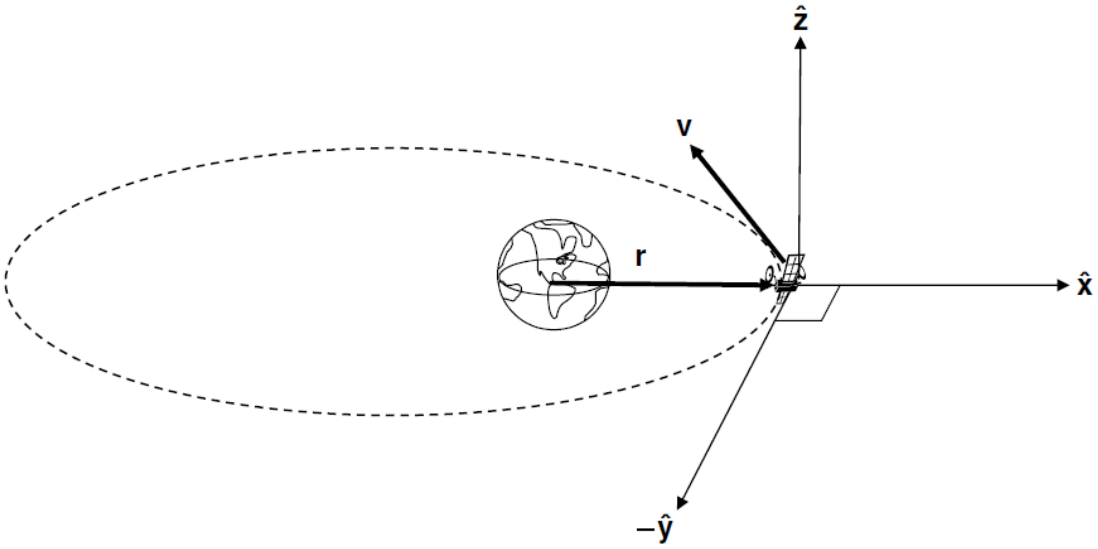


Figure 3: LVLH Frame[12]

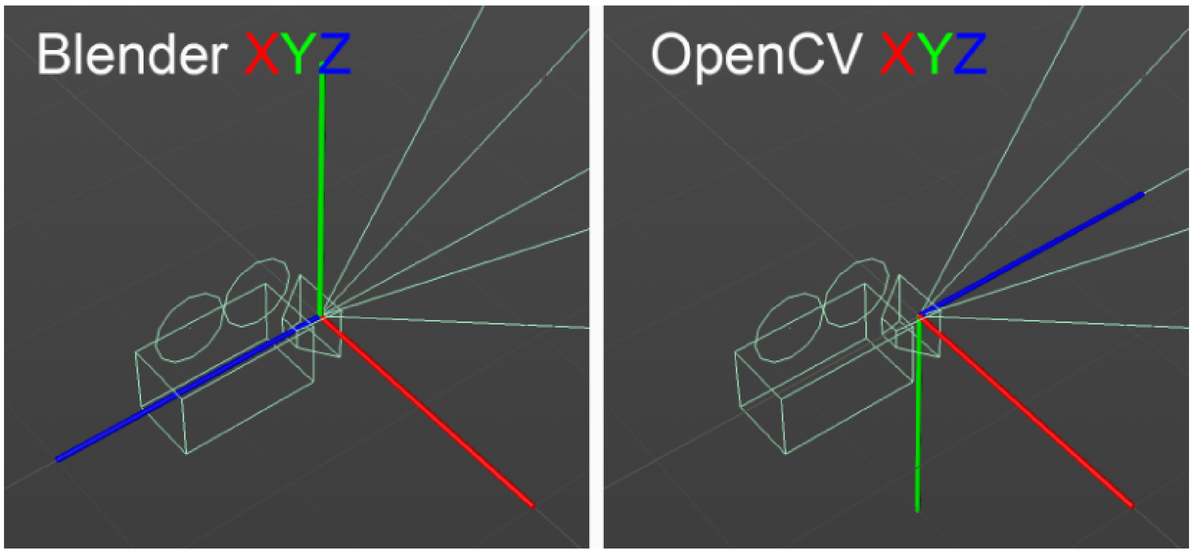


Figure 4: Blender and OpenCV Frame[2]

2.2. Chaser Orbital Dynamics

Assuming there are only two objects in space and the gravitational field is the only interaction between them, we get the following equation for keplerian orbits[18]:

$$\ddot{\mathbf{r}} = -\mu \frac{\mathbf{r}}{r^3}$$

Any effect that makes the object to deviate from the Keplerian orbit is considered as a perturbation and the perturbations usually considered are atmospheric drag, solar radiation pressure, nonspherical central body and the above equation is modified as follows to include the perturbations[18]:

$$\ddot{\mathbf{r}} = -\mu \frac{\mathbf{r}}{r^3} + \mathbf{p}$$

The vector **p** includes all other perturbations other than the spherical gravitational field contribution. The above equation is integrated numerically to obtain the predicted position **r** and velocity **v** given the initial conditions (r_0,v_0).

2.2.1. Gravitational Perturbations

Many of the spinning celestial bodies are not perfectly spherical in shape and can be considered as oblate spheroids. The gravitational field for such a planet varies with latitude and radius because of the equatorial bulge. Spherical harmonics is used to derive the formula for the perturbing acceleration due to gravitational effects. J_2 is the largest zonal harmonic and to simplify the model, the contribution due to J_2 to the perturbing acceleration is considered and others are ignored[18]. The perturbing gravitational acceleration $\mathbf{p}$ due to J_2 is given by:

$$\mathbf{p} = \frac{3}{2}\frac{J_2 \mu R^2}{r^4}\left[\frac{x}{r}\left(5\frac{z^2}{r^2} - 1\right)\hat{\mathbf{i}} + \frac{y}{r}\left(5\frac{z^2}{r^2} - 1\right)\hat{\mathbf{j}} + \frac{z}{r}\left(5\frac{z^2}{r^2} - 3\right)\hat{\mathbf{k}}\right]$$

2.2.2. Atmospheric Drag

For Earth, altitude above 100 km is considered as outer space and the air density at this altitude is still strong enough to alter the spacecraft motion. The drag will reduce the velocity and altitude of the spacecraft and it can eventually crash to the Earth's surface if not burned up in the atmosphere. There are several models that describe the atmospheric properties and a simple model is the US Standard Atmosphere 1976 and an exponential interpolation is used to obtain the intermediate values[18]. The perturbing acceleration due to drag is given by:

$$\mathbf{p} = -\frac{1}{2}\rho v_{rel}\left(\frac{C_D A}{m}\right)\mathbf{v}_{rel}$$

2.3. Chaser Attitude Equations of Motion

The equations of rigid body motion can be used to study the attitude dynamics of spacecraft. In this section, we will discuss the disturbance torques acting on the spacecraft and the sensors and actuators models used to control the angular velocity and attitude of the chaser spacecraft. The disturbance torques considered are gravity gradient torque, atmospheric drag torque and magnetic torque. Gyro and star sensors are used as sensors and reaction wheels are used as actuators.

2.3.1. Disturbance Torques

Gravity Gradient Torque

The gravity gradient torque is caused by the gravitational force acting on the spacecraft body. It's impact on spacecraft with non-symmetric body is much higher. The gravity gradient torque is given by[26]:

$$\mathbf{N}_{gg} = \frac{3\mu}{R^3}\left[\mathbf{R} \times (\mathbf{IR})\right]$$

where $\mathbf{R}$ is the position vector of the spacecraft with respect to the Earth, $\mathbf{I}$ is moment of inertia matrix and μ is Earth's gravitational constant.

Atmospheric Drag Torque

The aerodynamic drag torque is caused by the atmosphere around the spacecraft resisting it's motion. Atmospheric density and area of the spacecraft interacting with the atmosphere influences the amount of torque generated. The magnitude of the torque increases exponentially as the altitude decreases and hence is an important disturbance to be accounted for low earth orbits. It acts in the opposite direction of the velocity for the spacecraft[26]. The atmospheric drag torque is given by:

$$\mathbf{N}_{aero} = \mathbf{F}_{aero} \times \mathbf{R}_{com}$$

with

$$\mathbf{F}_{aero} = -\frac{1}{2}\rho C_D \mathbf{v}^2(\hat{\mathbf{N}}.\hat{\mathbf{v}})\hat{\mathbf{v}}dA$$

where C_D is drag coefficient, ρ is density, v is velocity, dA is exposed area.

Magnetic Torque

The magnetic torque is generated by the interaction of Earth's magnetic field with the magnetic field produced within the spacecraft. The magnetic dipole of spacecraft structure is the major contributor for this torque[26]. The magnetic torque is given by:

$$\mathbf{N}_{mag} = \mathbf{m} \times \mathbf{B}$$

where $\mathbf{m}$ is the magnetic moment of spacecraft and $\mathbf{B}$ is the Earth's magnetic field.

2.3.2. Sensors

Gyroscope

A rate gyroscope measures the angular velocity of the spacecraft in it's body frame. These measurements are affected by bias and white noise and the simple gyro model is given as:

$$\bar{\boldsymbol{\omega}} = \boldsymbol{\omega} + \mathbf{b} + \boldsymbol{\eta}_g$$

where $\mathbf{b}$ is the bias and $\boldsymbol{\eta}_g$ is the Gaussian white noise.

Star Sensor

Star sensors are one of the most accurate attitude determination sensors available and it is modelled as a typical measurement as follows:

$$\bar{\mathbf{e}} = \mathbf{e} + \mathbf{b} + \boldsymbol{\eta}_s$$

where $\mathbf{e}$ is euler angles, $\mathbf{b}$ is the euler angle bias and $\boldsymbol{\eta}_s$ is the Gaussian white noise.

2.3.3. Actuators

Reaction Wheels

Reaction wheels affect the inertia of the body and must be modelled before the attitude dynamics. Three reaction wheels with one on each axis and with it's own angular velocity ω_{Ri} and angular acceleration α_{Ri} is implemented for the chaser spacecraft. The total angular momentum of the chaser spacecraft is given by the following equation where $\vec{\omega}_{B/I}$ is the angular velocity of the chaser spacecraft[32].

$$\vec{H}_S = I_B \vec{\omega}_{B/I} + \sum_{i=1}^{3} I_{Ri}^B \omega_{Ri} \hat{n}_{Ri}$$

where $\hat{n}_{Ri}$ denotes the reaction wheel's rotation axis and I_{Ri}^B denotes the reaction wheel inertia matrix.

A simple PD control is used and the desired torque to be placed on spacecraft can be computed by:

$$\vec{M}_{desired} = -k_p(\epsilon_i - \epsilon_{desired}) - k_d(\omega_i - \omega_{desired})$$

The above equation is equated to the equation of torque placed on spacecraft as:

$$\vec{M}_{desired} = \vec{M}_R = \sum_{i=1}^{NR} I_{Ri}^B \alpha_{Ri} \hat{n}_{Ri} = \mathbf{J}\vec{\alpha}$$

The angular acceleration of the reaction wheel is then computed as follows:

$$\vec{\alpha} = \left(\mathbf{J}^T \mathbf{J}\right)^{-1} \mathbf{J}^T \vec{M}_{desired}$$

2.3.4. Complimentary Filter

If $\mathbf{q}$ is the covariance of model noise and $\mathbf{r}$ is the covariance of the measurement noise, then the filter is designed as[32]:

$$\tilde{x} = (\mathbf{1} - \mathbf{s})\bar{y} + \mathbf{s}\tilde{x}^{-}$$

where $\mathbf{q}^{-1} = \mathbf{s}$ and $\mathbf{r}^{-1} = \mathbf{1} - \mathbf{s}$ and $\tilde{x}^{-}$ is the apriori estimate.

When $\mathbf{s} = \mathbf{1}$, the new and the previous estimate are the same and that the sensor cannot be relied upon. When $\mathbf{s} = 0$, the new estimate matches the sensor measurement and that the model noise cannot be relied upon. It is a first order filter and can be used when the knowledge of covariance is high.

2.3.5. Kinematic and Dynamic Model

The derivative of angular velocity can be found by using Euler's equations of motion and the reaction wheel are also included in that equation.

$$\dot{\vec{\omega}}_{B/I} = \mathbf{I}^{-1}[\mathbf{N} - (\boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega})]$$

where $\mathbf{N}$= $\mathbf{N}_{ext}$-$\mathbf{N}_{RW}$. $\mathbf{N}_{ext}$ is the external disturbance torques acting on spacecraft and $\mathbf{N}_{RW}$ is the torque produced by the reaction wheel.

The attitude is propagated using the following equation and quaternion is used to represent the attitude.

$$\begin{Bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{Bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{Bmatrix}$$

where q_i are the quaternions and $p, q,\ r$ are angular velocity vector in body frame.

2.4. Relative Spacecraft Motion Model

2.4.1. Relative Translational Model

The relative dynamics is described by Clohessy-Wiltshire equations and it holds for nearly circular orbits and the relative distance between target and chaser less than the radius of the orbit.

$$\ddot{x} = 3n^2x + 2n\dot{y}$$

$$\ddot{y} = -2n\dot{x}$$

$$\ddot{z} = -n^2z$$

where x-axis is along radius vector, z-axis along angular momentum vector and y-axis completes the frame. The closed form solution for these equations are given by:

$$x(t) = (4 - 3\cos nt)x_0 + \frac{\sin nt}{n}\dot{x}_0 + \frac{2}{n}(1 - \cos nt)\dot{y}_0$$

$$y(t) = 6(\sin nt - nt)x_0 + y_0 - \frac{2}{n}(1 - \cos nt)\dot{x}_0 + \frac{4\sin nt - 3nt}{n}\dot{y}_0$$

$$z(t) = z_0\cos nt + \frac{\dot{z}_0}{n}\sin nt$$

where

$$n = \sqrt{\frac{\mu}{a^3}}$$

and a is the semi-major axis of the spacecraft.

2.4.2. Relative Rotational Model

The relative rotational dynamics is given in the chaser frame as follows[12]:

$$I_0\dot{\omega} = I_0DI_1^{-1}\left[N_1 - D^T(\omega + \omega_0) \times I_1D^T(\omega + \omega_0)\right] - I_0\omega_0 \times \omega - [N_0 - \omega_0 \times I_0\omega_0]$$

where the subscript 0 represents the chaser spacecraft values, subscript subscript 1 represents target spacecraft values and D is the rotation matrix from target to the chaser. The rotation matrix $D(\beta)$ is given below:

$$\mathbf{D}(\boldsymbol{\beta}) = \begin{bmatrix} {\beta_1}^2 - {\beta_2}^2 - {\beta_3}^2 + {\beta_0}^2 & 2(\beta_1\beta_2 - \beta_3\beta_0) & 2(\beta_1\beta_3 + \beta_2\beta_0) \\ 2(\beta_1\beta_2 + \beta_3\beta_0) & -{\beta_1}^2 + {\beta_2}^2 - {\beta_3}^2 + {\beta_0}^2 & 2(\beta_2\beta_3 - \beta_1\beta_0) \\ 2(\beta_1\beta_3 - \beta_2\beta_0) & 2(\beta_2\beta_3 + \beta_1\beta_0) & -{\beta_1}^2 - {\beta_2}^2 + {\beta_3}^2 + {\beta_0}^2 \end{bmatrix}$$

The relative quaternion can be propagated directly from the kinematic relation as follows:

$$\begin{Bmatrix} \dot{\beta}_0 \\ \dot{\beta}_1 \\ \dot{\beta}_2 \\ \dot{\beta}_3 \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \begin{Bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{Bmatrix}$$

where ω components are relative angular velocity values.

3 | Observation Model

3.1. Pinhole Camera Model

In computer vision, there are several models which explain the working of optical camera. In this section, we describe the pinhole camera model which is a simple model for the camera which gives accurate results. In this concept, any point in the environment can send a single light ray into the pinhole. The size of the image in relation to the far-off object is determined by just one camera parameter called focal length after this point is "projected" onto an image plane that is always in focus[34].

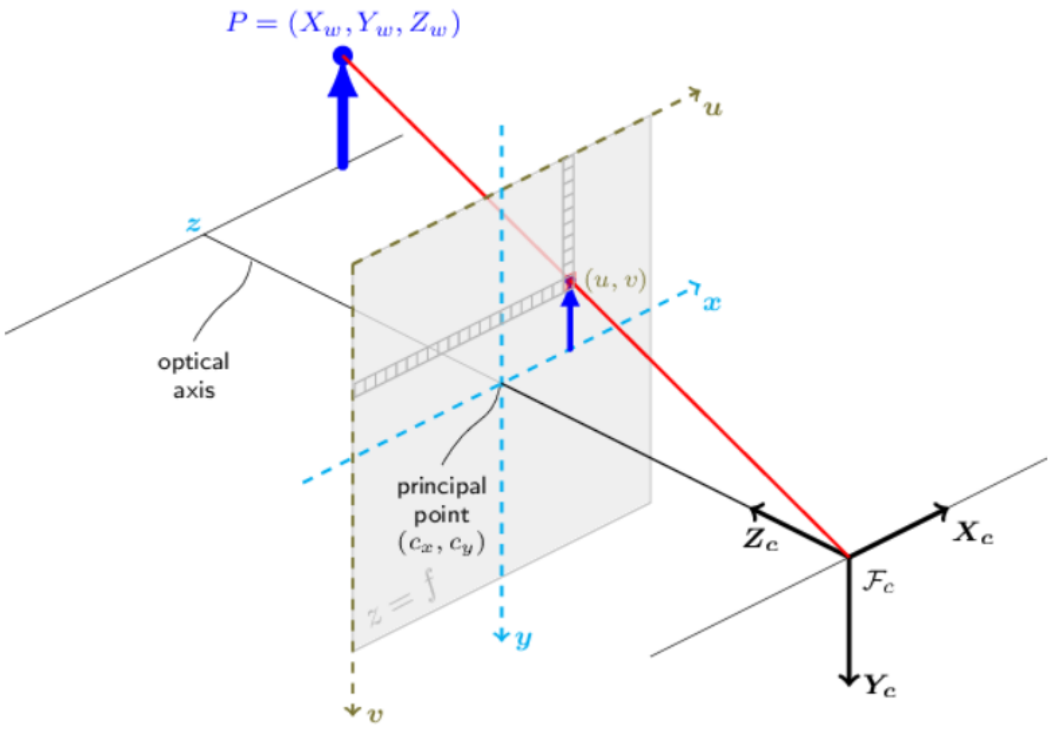


Figure 5: Pinhole Camera Model[7]

The focal length is determined by measuring the distance between the image plane and the pinhole camera aperture. Consequently, the following equations can be used to project object points into image points:

$$u = -f\frac{X_c}{Z_c}$$

$$v = -f\frac{Y_c}{Z_c}$$

The pinhole camera model can now be rearranged in an equivalent form that is simpler mathematically. This can be done by switching the pinhole and picture plane, so that the object now appears to be right side up. The center of projection now stands in for the former pinhole point. Each ray that emanates from the object in this perspective is pointed toward the center of projection and intersects the image plane. The primary point is the location where the optical axis and the picture plane converge. The picture of the distant object in this updated version of the pinhole camera model is the same size as it was on the image plane in the original version. The item image is no longer upside down, hence the negative sign has vanished[34].

The center of the imager chip must be included, and since it is not on the optical axis, two new parameters, c_x and c_y, must be introduced to describe a potential displacement of the center of coordinates on the image plane with respect to the optical axis. As a result, the following equations may be used to create a very straightforward model in which a point **P** in the real world with coordinates of (X_w, Y_w, Z_w) is projected onto the image plane at some pixel location given by (u,v)[7, 34]:

$$u = f_x\frac{X_c}{Z_c} + c_x$$

$$v = f_y\frac{Y_c}{Z_c} + c_y$$

Two distinct focal lengths f_x and f_y have been introduced. Additionally, the focal length f_x (as well as f_y) is actually the result of the physical focal length (typically measured in millimeters units) of the lens used by the camera and the size (typically measured in pixels per millimeter units) of the individual imager elements, meaning that f_x unit is in pixels. The size of the imager elements and the physical focal length cannot be determined directly by any camera calibration technique; instead, only combinations of these parameters may be derived without actually disassembling the camera and measuring its individual parts.

Projective transform refers to the relationship that converts a collection of points **P** in the real world with coordinates (X_w, Y_w, Z_w) to points projected on the image with coordinates (u,v). It is practical to utilize the well-known homogeneous coordinates for this type of change. With the additional restriction that any two points whose values are proportional are actually comparable points, these coordinates associated with a point in a projective space of dimension n are commonly written as a (n+1)-dimensional vector[34]. Since the image plane in this application is the projective space, the two-dimensional image coordinates are now represented by a three-dimensional vector, $\vec{p}$ = (u,v,w). By dividing through by w, keeping in mind the additional limitation, the pixel coordinates can be obtained. The camera-defining characteristics can be rearranged in a single 3×3 matrix (A), also known as the camera intrinsic matrix, thanks to the homogeneous coordinates. The following short form now describes how the item points in the real world are projected onto the camera:

$$\vec{p} = A\vec{P}$$

where $\vec{p} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$, $A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ and $\vec{P} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$

3.1.1. Camera Intrinsic Parameters

The camera's intrinsic parameters are those that are most directly related to its physical properties. Some of them, like the focal lengths (f_x and f_y) and the optic center (c_x, c_y), which make up the camera intrinsic matrix A, have already been discussed by the pinhole model. There are additional lens distortion-related characteristics which will be defined but are not used in the model.

There are primarily two types of lens distortions: tangential distortion results from the construction of the camera, while radial distortion results from the curvature of the lens. The positioning of pixels close to the boundaries of the image plane is frequently distorted by the lenses of real cameras. The "fisheye" effect is a result of this phenomena.

There are five parameters that correct lens distortion and these factors are typically combined into a distortion vector, which makes it possible to convert the raw pixel coordinates Q into the right coordinates that most closely match the 3D point with its image-plane projection P[7, 34].

3.1.2. Camera Extrinsic Parameters

The relationship between the camera reference system and another arbitrary system is covered by the extrinsic parameters. In actuality, there hasn't been any discussion of the physical world's object reference system up to this point. Since the preceding relationship was established, $P = (X_w, Y_w, Z_w)$ corresponds to the projection's center. The world coordinate reference system is a distinct Euclidean coordinate frame that is typically used to express points in space. Translation and a rotation are used to connect the two coordinate frames considered.

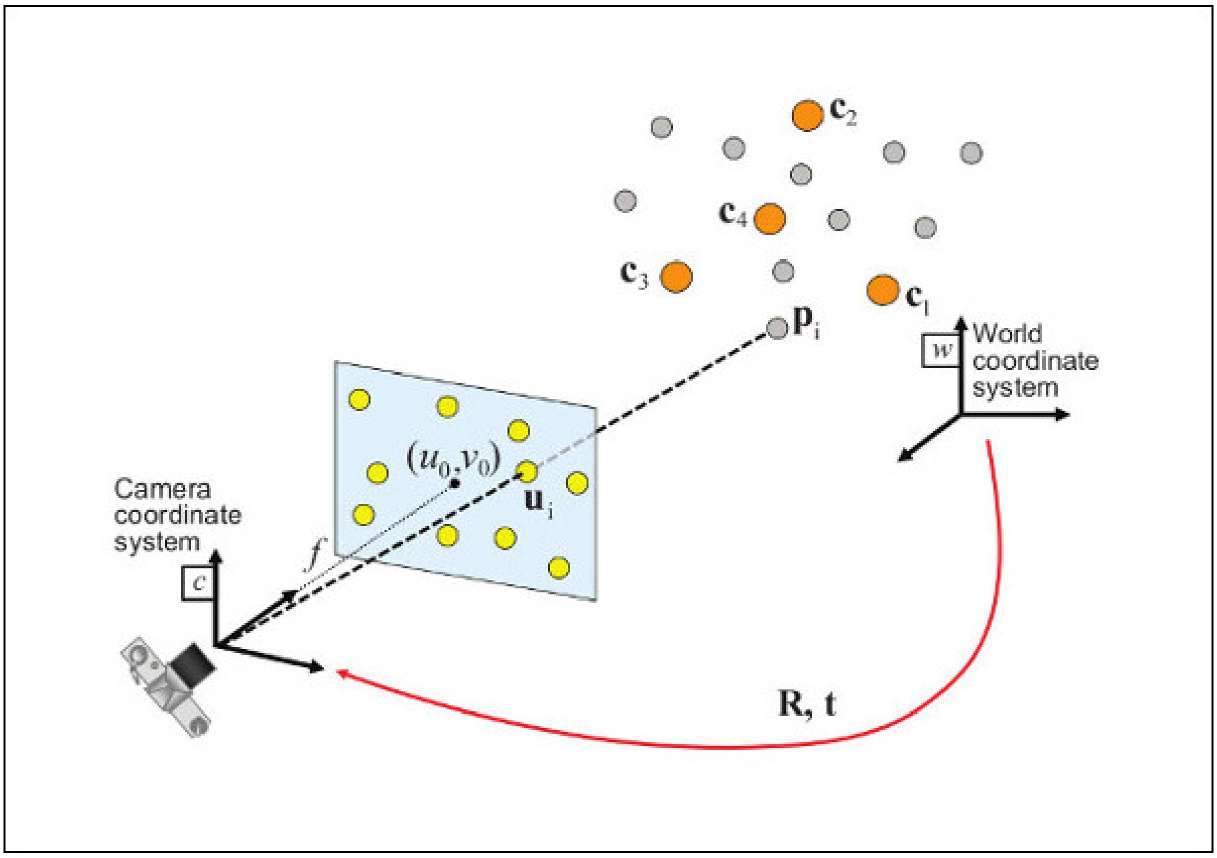


Figure 6: World and Camera Coordinate frame Relation[19]

The translation of the two systems causes the camera's origin and the coordinate frame for the object to coincide. In this application, it is the actual distance between the object point and the camera's center of projection. It stands for the offset between the two origins. It enables a comparable representation of a point position in a different reference frame when the angle is rotated. The rotation in the three-dimensional space needed for these two systems can be broken down into three separate two-dimensional rotations, each centered around one of the x, y, and z axes.

3.1.3. Formulation and Solution

The homogeneous transformation represents the shift in basis from the world coordinate system (w) to the camera coordinate system (c), and it is conveyed by the extrinsic parameters R and t. In order to acquire the representation of the point P in the camera coordinate system, P_c, given its representation in world coordinates, P_w, we use[7]:

$$P_c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} P_w$$

R, a 3-by-3 rotation matrix, and t, a 3-by-1 translation vector, make up this homogeneous transformation:

$$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and therefore:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Combining the equations for intrinsics and extrinsics, we can formulate

$$sp = A[R|t]P_w$$

as

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

3.2. ArUco Markers

A synthetic square marker known as an ArUco marker consists of a large black border and an inside binary matrix that serves as the marker's identifier (id). Additionally, it is feasible to estimate the camera pose, which entails determining the position and orientation of the camera in relation to the markers, if the camera has been calibrated, meaning that its intrinsic properties are known.

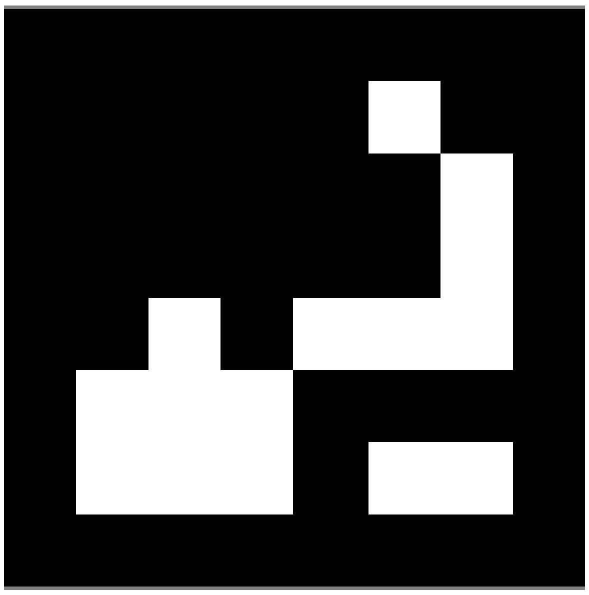


Figure 7: Example of ArUco Marker

ArUco markers come in a variety of categories, each of which belongs to a dictionary. The amount of markers and inner squares that encode the binary pattern varies amongst the dictionaries.

The benefits of these markers, in addition to their quick and reliable identification, include the fact that each one offers a 4-point vector (representing the image's corners) and a special ID that distinguishes them based on their pattern and the dictionary to which they belong. The 4-point vector's corner pixel coordinates are also arranged clockwise, beginning with the upper-right corner. This information provides a marker with its related reference system.

The detection procedure must produce a list of identified markers when given an image with ArUco markers. Each found marker consists of: its four corners' placement in the picture (in their original order) and the marker's identifier[1]. There are two basic steps in the marker detecting process:

1. Finding potential marker candidates: This stage involves analyzing the image to identify potential marker possibilities, such as square shapes. The markers are first segmented using adaptive thresholding. Next, contours are recovered from the thresholded image, and any that are not convex or roughly square in shape are removed[1].

2. By examining their internal codification, it is crucial to ascertain whether the candidates found during candidate detection are in fact markers. The first stage in this

process is to separate each marker's marker bits. To do this, the marker's canonical form is first obtained by applying a perspective transformation. After that, Otsu thresholding is used to separate the white and black bits in the original image. In accordance with the size of the border and the markers, the image is divided into several cells. Then, to establish if a cell contains a white or a black bit, the number of black or white pixels in each cell is tallied. The bits are finally examined to establish whether the marker belongs to the particular dictionary. Techniques for correcting errors are used as necessary[1].

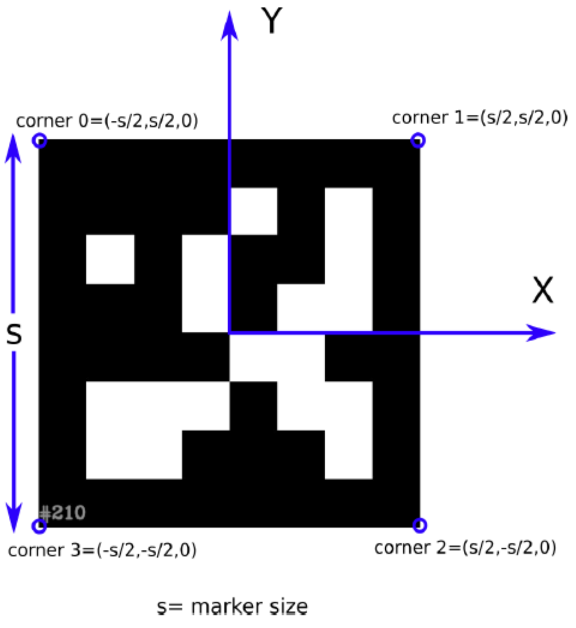


Figure 8: ArUco Relative Coordinate System[34]

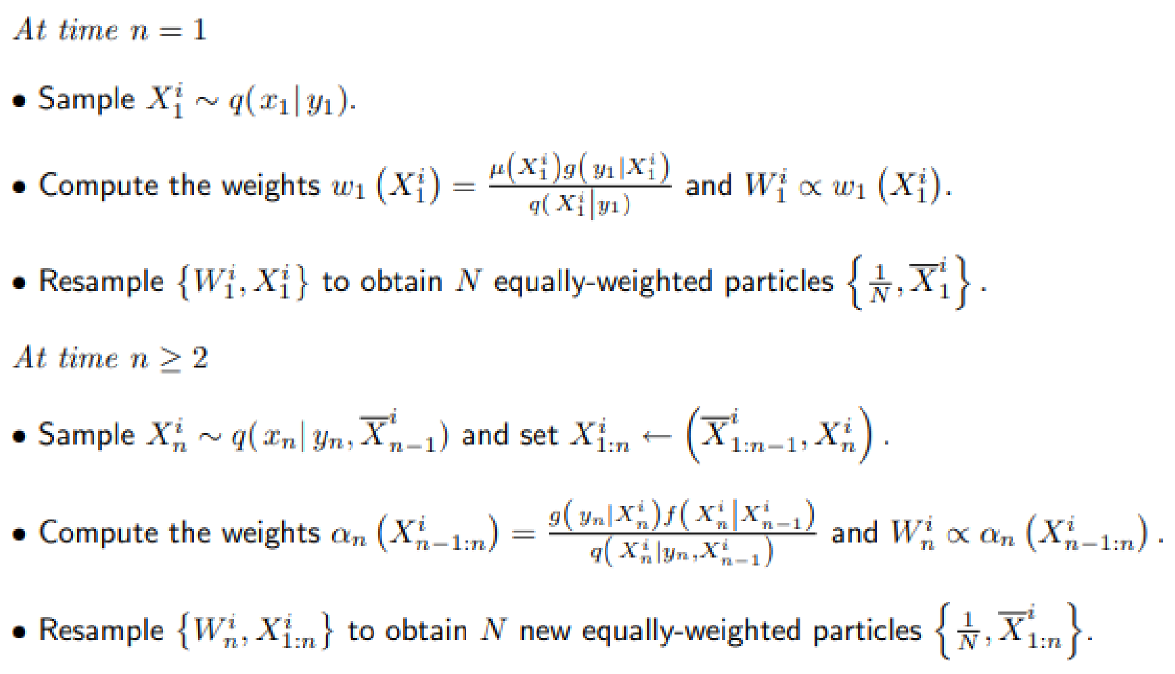
After finding the markers, you'll presumably want to get their camera position as your next move.

You must be aware of your camera's calibration parameters in order to do camera pose estimation. These are the distortion coefficients and camera matrix. With ArUco markers, you can individually estimate each marker's pose while estimating the pose. The 3D transformation from the marker coordinate system to the camera coordinate system determines the camera pose in relation to a marker. Vectors for rotation and translation define it. A function to estimate the postures of all the observed markers is offered by the ArUco module[1].

4 | Estimation Methodology

4.1. Particle Filter

Particle filters are a widely prominent class of numerical approaches for the resolution of optimal estimation problems in non-linear non-Gaussian settings since its inception in 1993[23]. The main benefit of particle approaches over conventional approximation techniques, such as the well-known Extended Kalman Filter, is that they don't rely on any local linearization methods or rudimentary functional approximations. These methods are computationally expensive, thus there is a cost associated with this versatility. However, similar techniques are already applied in real-time applications in a variety of domains, including robotics, financial econometrics, computer vision, target tracking, and chemical engineering, because of the availability of ever-increasing computational capacity.



At time $n = 1$

- Sample $X_1^i \sim q(x_1 | y_1)$.
- Compute the weights $w_1\left(X_1^i\right) = \frac{\mu(X_1^i) g(y_1 | X_1^i)}{q(X_1^i | y_1)}$ and $W_1^i \propto w_1\left(X_1^i\right)$.
- Resample $\{W_1^i, X_1^i\}$ to obtain N equally-weighted particles $\left\{\frac{1}{N}, \overline{X}_1^i\right\}$.

At time $n \geq 2$

- Sample $X_n^i \sim q(x_n | y_n, \overline{X}_{n-1}^i)$ and set $X_{1:n}^i \leftarrow \left(\overline{X}_{1:n-1}^i, X_n^i\right)$.
- Compute the weights $\alpha_n\left(X_{n-1:n}^i\right) = \frac{g(y_n | X_n^i) f(X_n^i | X_{n-1}^i)}{q(X_n^i | y_n, X_{n-1}^i)}$ and $W_n^i \propto \alpha_n\left(X_{n-1:n}^i\right)$.
- Resample $\{W_n^i, X_{1:n}^i\}$ to obtain N new equally-weighted particles $\left\{\frac{1}{N}, \overline{X}_{1:n}^i\right\}$.

Figure 9: General Algorithm of Particle Filter[20]

All of the weights added together equal one. Without resampling, the particle filter would instead result in a collection of distinct simulations, each with a unique weight or probability. The so-called sample depletion would most likely take place. This indicates that, in the absence of input from observation, all weights would go toward zero, save for one that would be nearly equal to 1. The high value of weight just indicates that one sequence in the collection of particles is more likely than the others, not that the predicted state is nearer to the true one. The feedback from the observation is introduced through resampling, which also ensures that the accurate state estimation does not vanish[36].

Like the UKF, the PF approach has no restrictions on the noise distribution and does not rely on any local linearization techniques. Many instances where the EKF does not perform well (involving highly non linearities) can benefit greatly from this flexibility. But each of these benefits comes at a price. In actuality, the PF's computational load is its bottleneck[24].

Particle Filter function implemented in MatLab is used in this thesis. It requires to provide two functions namely, State Transition Function and Measurement Likelihood Function. The resampling policy can also defined in the particle filter object.

5 | Numerical Simulations

The chaser spacecraft states are propagated using Cowell's method and the initial orbital parameters are tabulated. The propagated position and velocity of the spacecraft for one orbital period is shown in the figure.

Orbital Elements	**Values**
Orbital Height	600 km
Eccentricity	0.001
Inclination	15°
AOP	340°
RAAN	0°
TA	0°

Table 1: Chaser Initial Orbital Elements

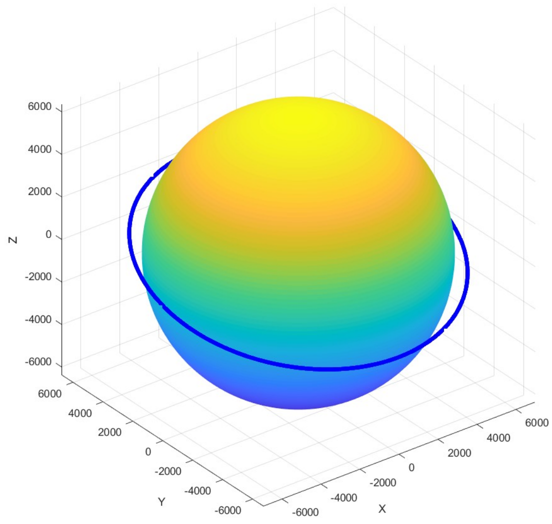


Figure 10: Chaser Orbit

Reaction wheels are used to control the spacecraft and we can see that the reaction wheels are working and the required attitude in euler angles [0 0 90°] and the required angular velocity of [0 0 0.1]°/s is obtained.

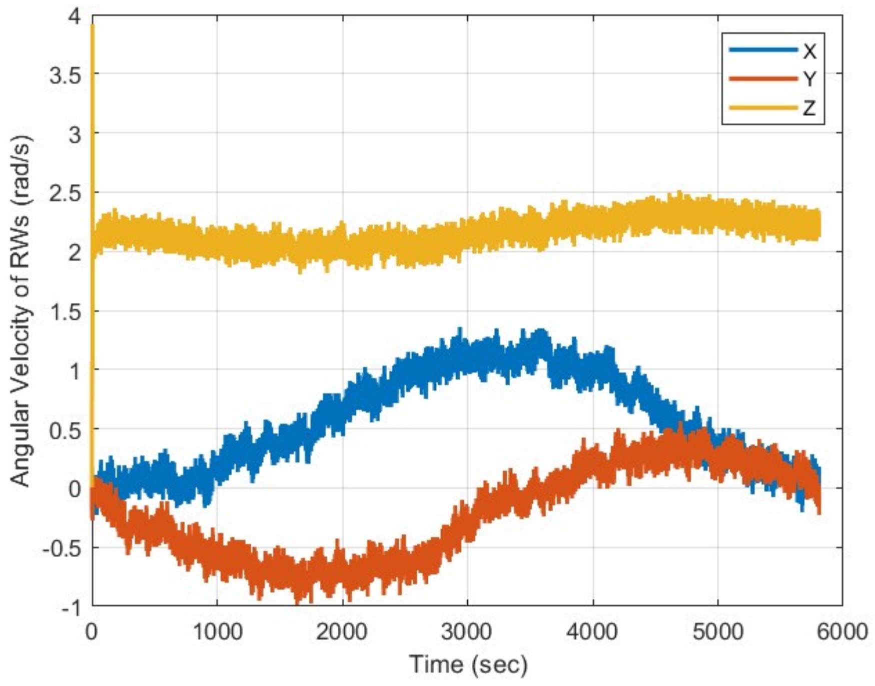


Figure 11: Angular Velocity of Reaction Wheels

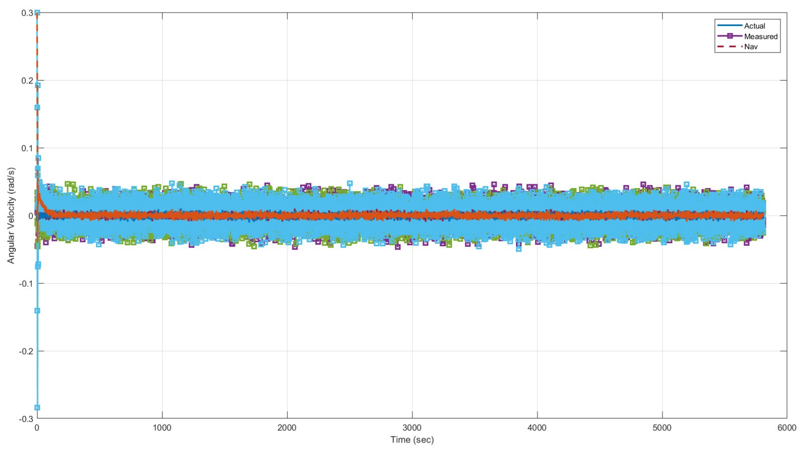


Figure 12: Angular Velocity of Chaser

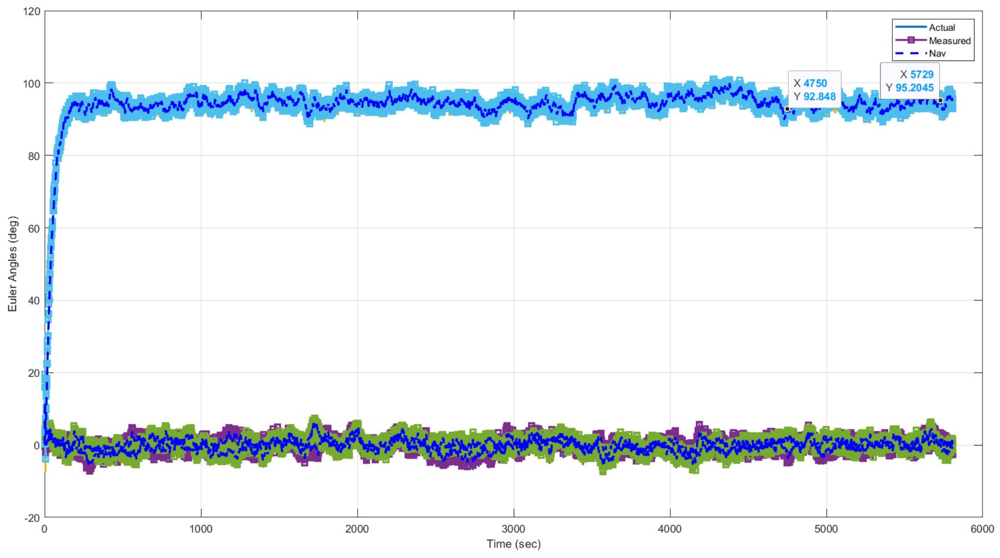


Figure 13: Euler Angles of Chaser

5.1. Relative Orbit

Clohessy–Wiltshire equations are used to propagate the relative motion of spacecraft and the initial conditions considered are [0 0 15] meters in relative position, [0 0 0.1]meters/second in relative velocity, [0.006 0.006 0.12] deg/second in relative angular velocity and relative quaternion is assumed as [1 0 0 0]. The plots obtained for the propagated states are given below.

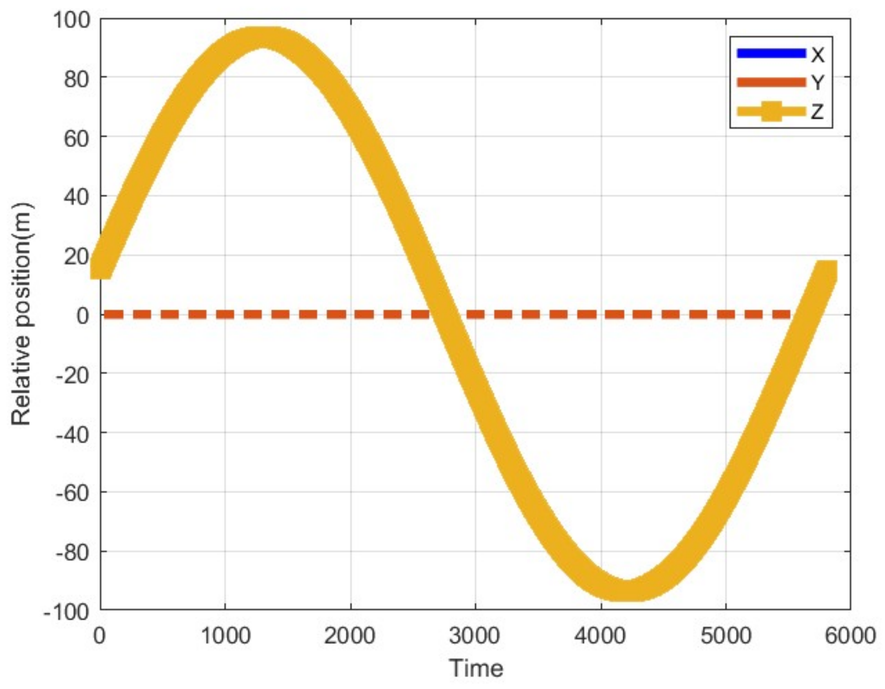


Figure 14: Relative Position in LVLH Frame

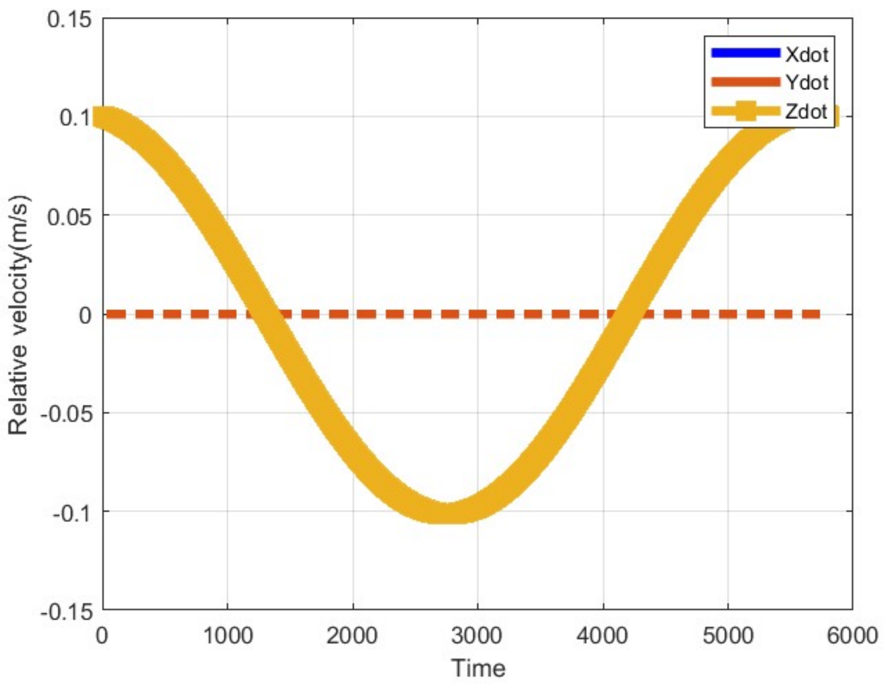


Figure 15: Relative Velocity in LVLH Frame

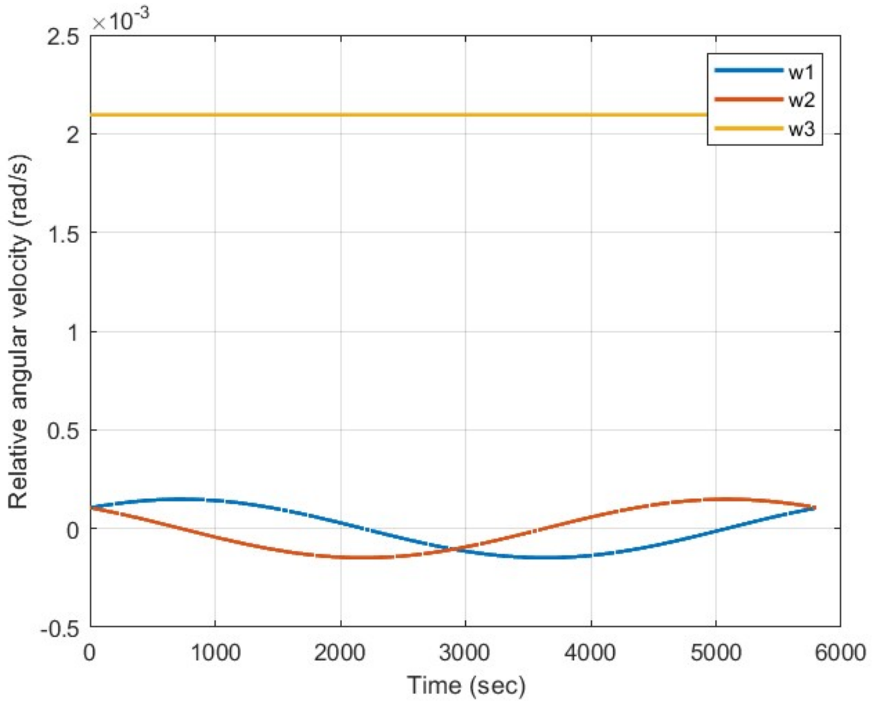


Figure 16: Relative Angular Velocity

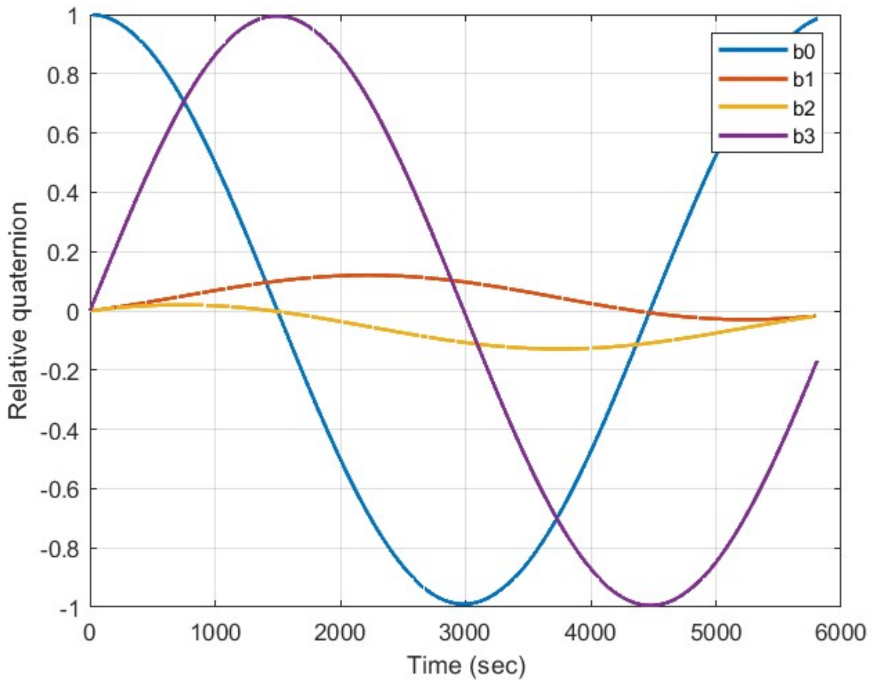
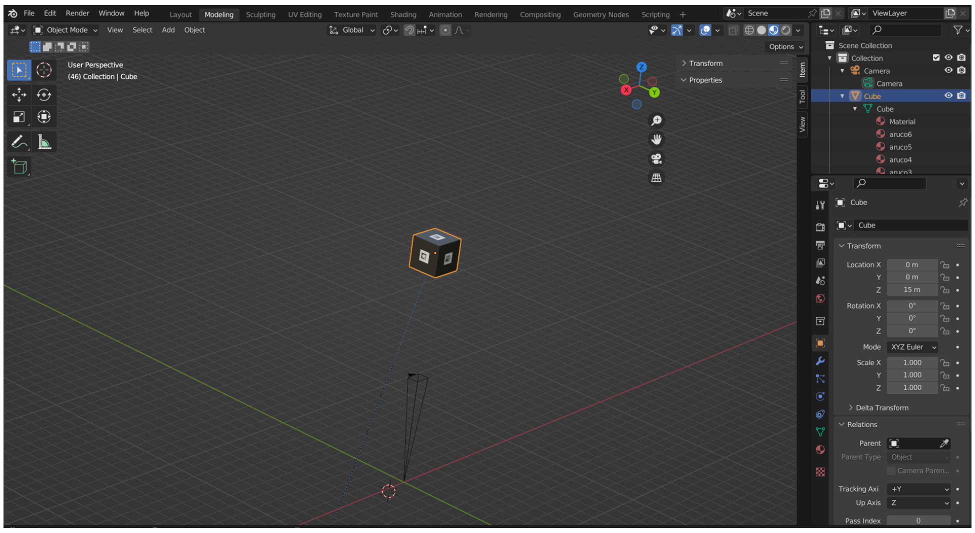


Figure 17: Relative Quaternion

5.2. Blender Setup

Blender is used to create the images that the chaser camera would view and used as the measurement for the filter. Blender has a camera which is set as the origin to mimic the chaser position. Then the relative parameters are given to the default cube in blender which is the target and thus we have the similar geometry as in real environment. The target spacecraft or 'Cube' has ArUco markers on the faces and 6 different ArUco markers were placed on all the faces of the cube.



Cube

Cube

Transform

Location X	0 m
Y	0 m
Z	15 m
Rotation X	0°
Y	0°
Z	0°
Mode	XYZ Euler
Scale X	1.000
Y	1.000
Z	1.000

Delta Transform

Relations

Parent	
Parent Type	Object
	Camera Paren...
Tracking Axi	+Y
Up Axis	Z
Pass Index	0

Figure 18: Blender Setup

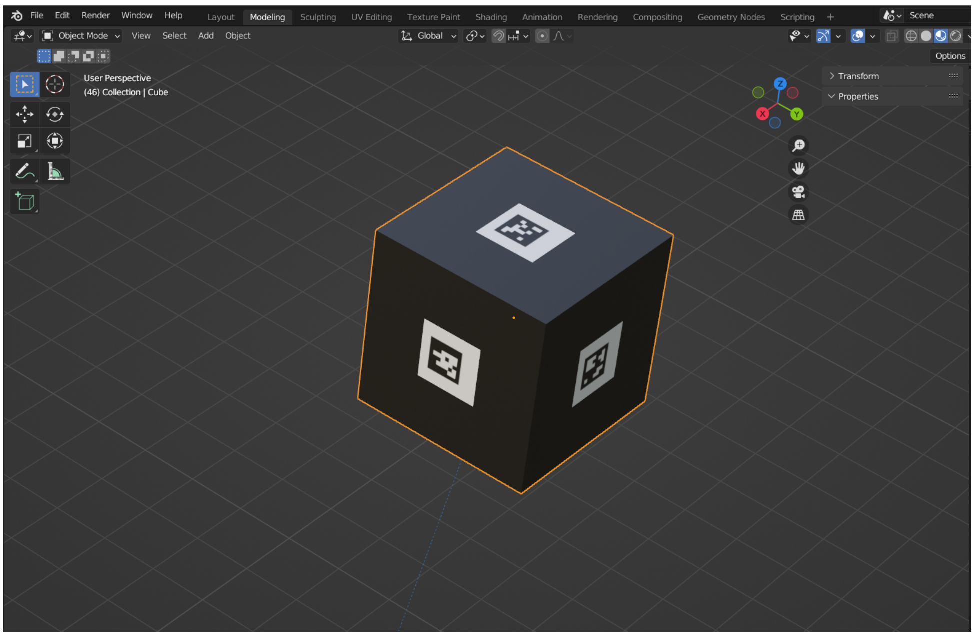


Figure 19: Target Spacecraft with ArUco markers

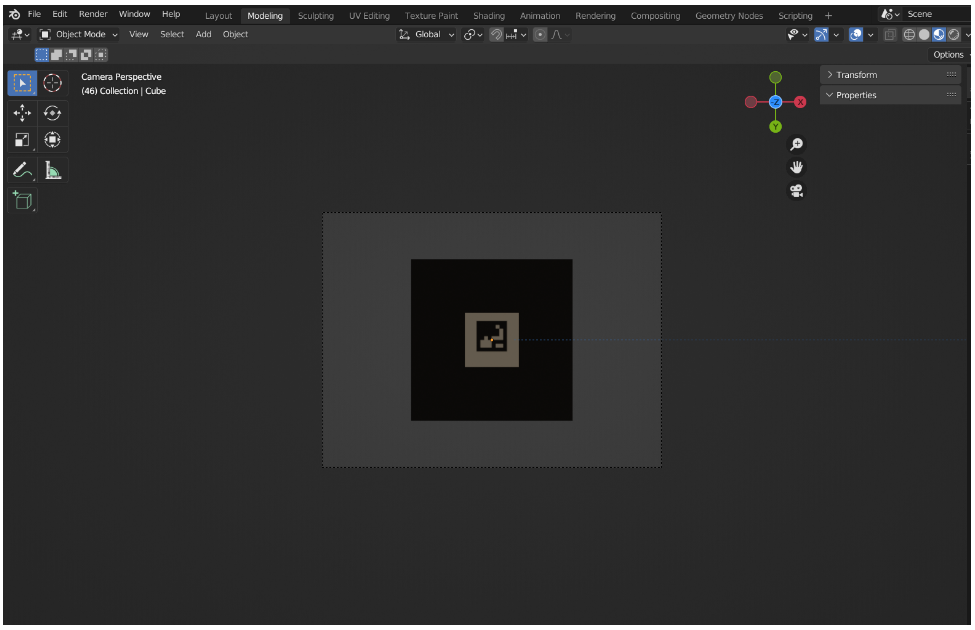


Figure 20: Target Spacecraft in Camera perspective

5.3. Measurement Model Validation

In order to validate the measurement model, the images are taken in as input and openCV functions in matlab are used to detect and estimate the position and orientation. This is compared with the true values input into blender to check if they are accurate. The blender and openCV camera reference frame is different and so rotation is required to be able to compare them.

The ArUco images are generated using *cv.drawMarkerAruco* and the predefined dictionary selected is $6 \times 6(250)$. The generated ArUco markers are given below.

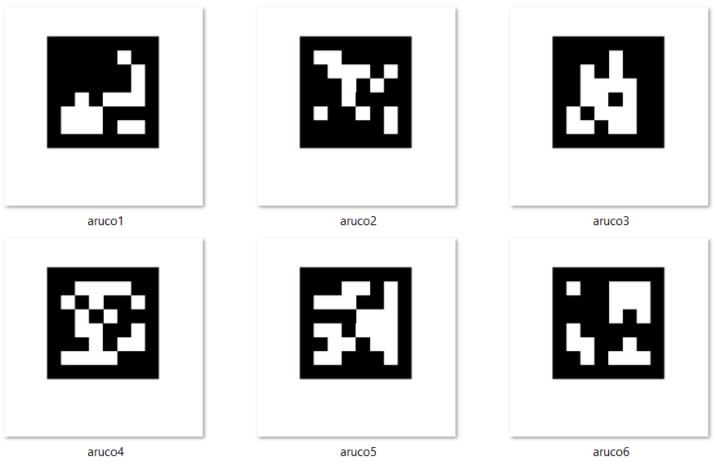


Figure 21: Generated ArUco Markers

The above ArUco markers are put on the sides of the cube. The *cv.detectMarkers* and *cv.drawDetectedMarkers* are used to detect and identify the markers from the predefined dictionaries. The identified image is given below:

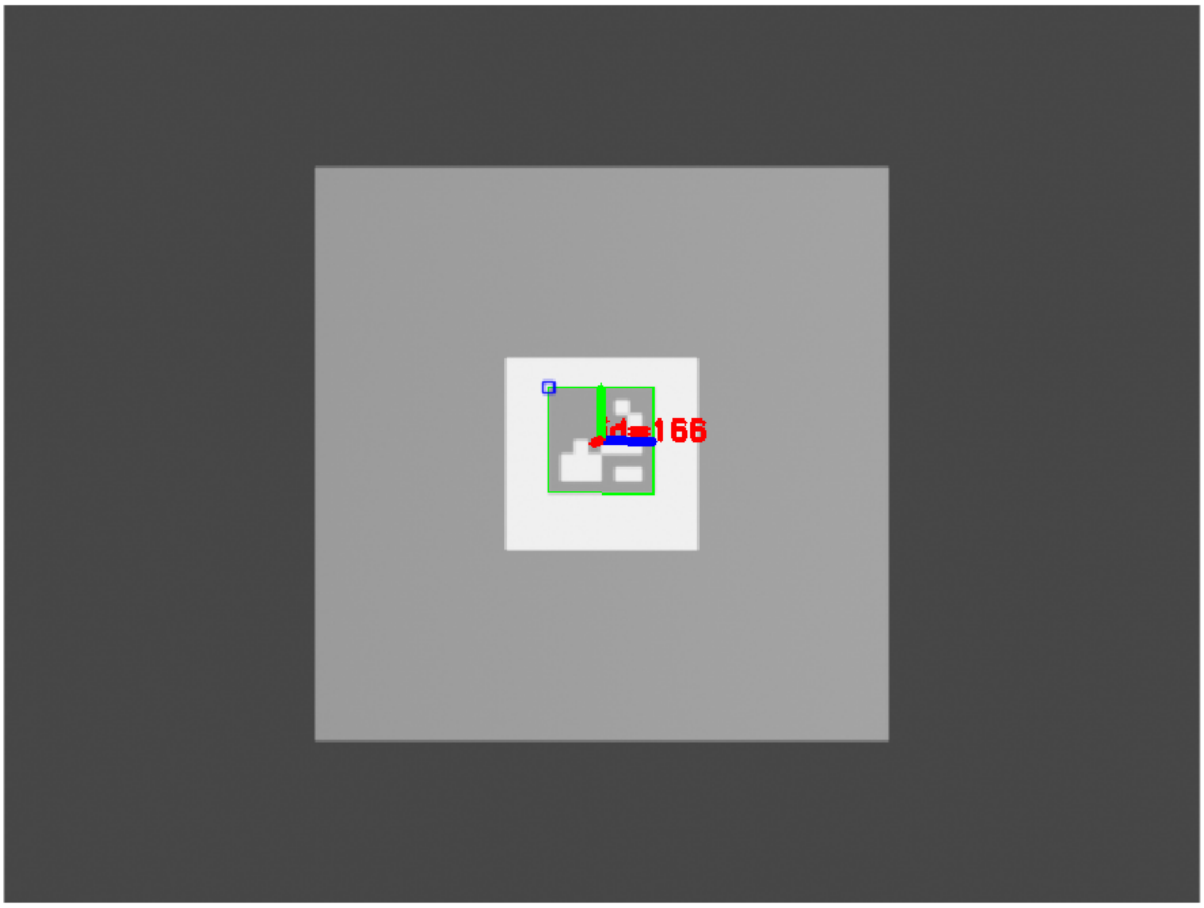


Figure 22: Detected and Estimated ArUco Marker

The camera calibration matrix was obtained through blender and the camera matrix is given by A. The focal length is considered as 250 mm and the units are in pixels for the A matrix.

$$A = \begin{bmatrix} 4444.4443359375 & 0 & 320 \\ 0 & 4444.4443359375 & 240 \\ 0 & 0 & 1 \end{bmatrix}$$

The radial and tangential distortion as not considered and *cv.estimatePoseSingleMarkers* is used to estimate the pose. This function returns the rotation and translation parameters and it is cross-checked with the true values. The position vector obtained through this estimation is [-0.0068;-0.0513;30.4359] meters and the true value is [0;0;30] meters. For the rotation, Blender camera was rotated [-180;0;0] to point at target and the obtained result is [-176.4404;-8.5880;0.0050]. This was multiplied by rotation matrix from opencv to blender system to obtain [3.5596;8.5880;-0.0050] and the true value was [0;0;0]. This validates to some extent the working of the measurement model and will require particle filter to obtain more accurate results.

5.4. Filter Dynamical Model Validation

In order to validate the filter dynamical model separately, the code was designed to neglect the measurements and take in only the predicted values. Hence there is no correction to the states involved. Also, the errors are eliminated in the initial states of the filter so that the errors don't build up and deviate from the true state and hence it can be validated if it matches with the true values. It can be seen from the plots that the true and the particle filter estimates match since there are no measurements and no errors.

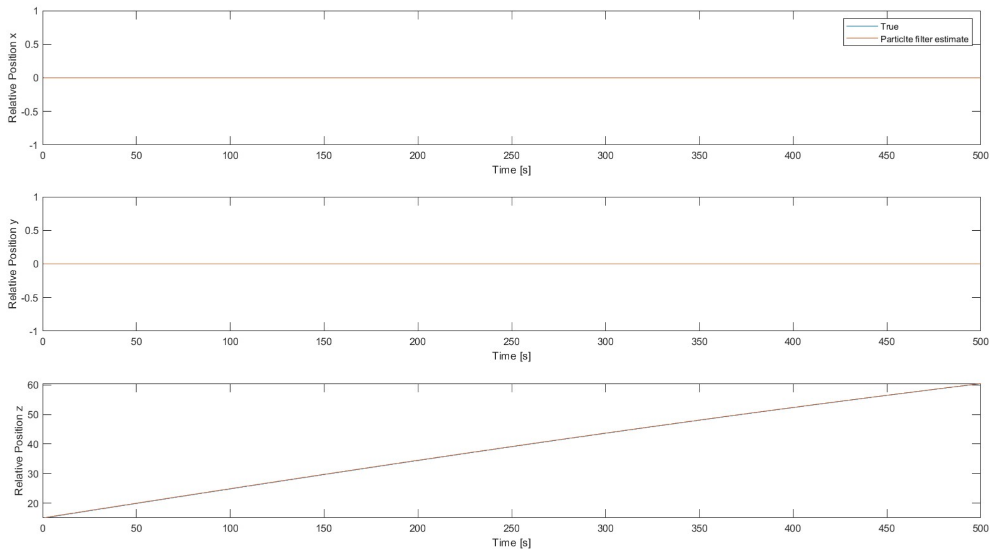


Figure 23: Particle Filter Dynamical Model Validation for Relative Position

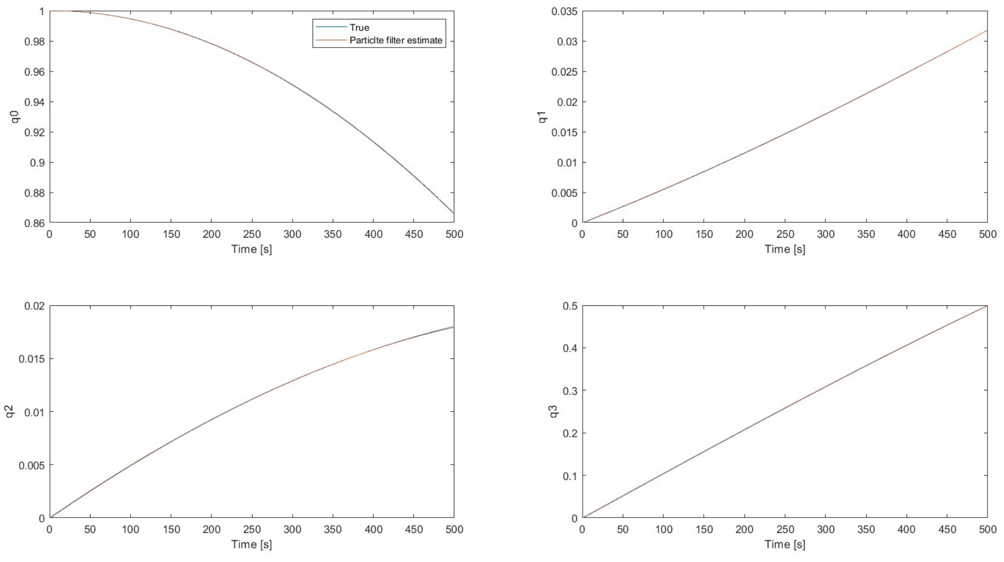


Figure 24: Particle Filter Dynamical Model Validation for Relative Quaternion

5.5. Particle Filter Run

The particle filter is run including the measurements and the plots of the true and estimated values and the error values are given below and the duration of the run is 500 seconds.

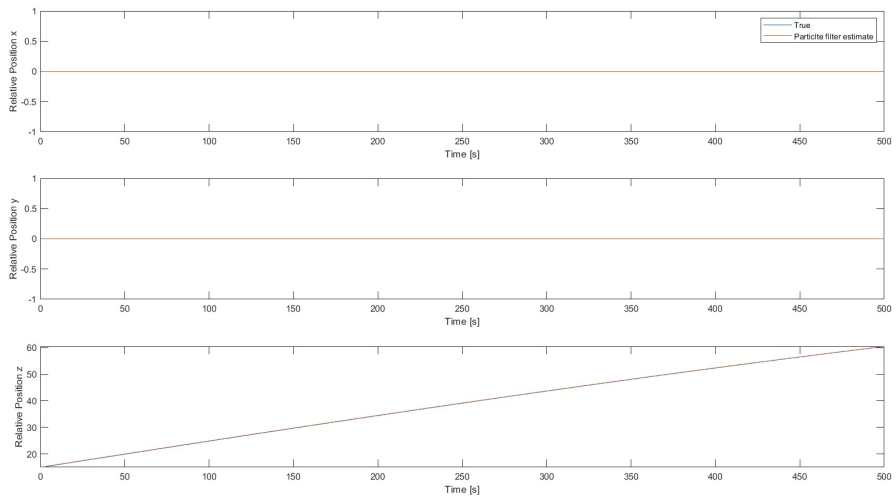


Figure 25: Particle Filter Estimate for Relative Position

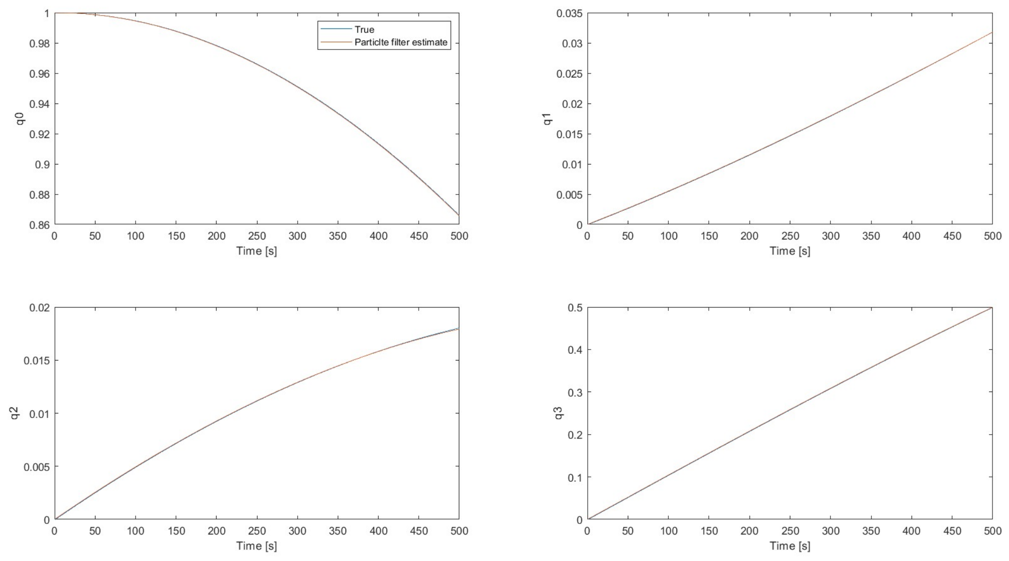


Figure 26: Particle Filter Estimate for Relative Quaternion

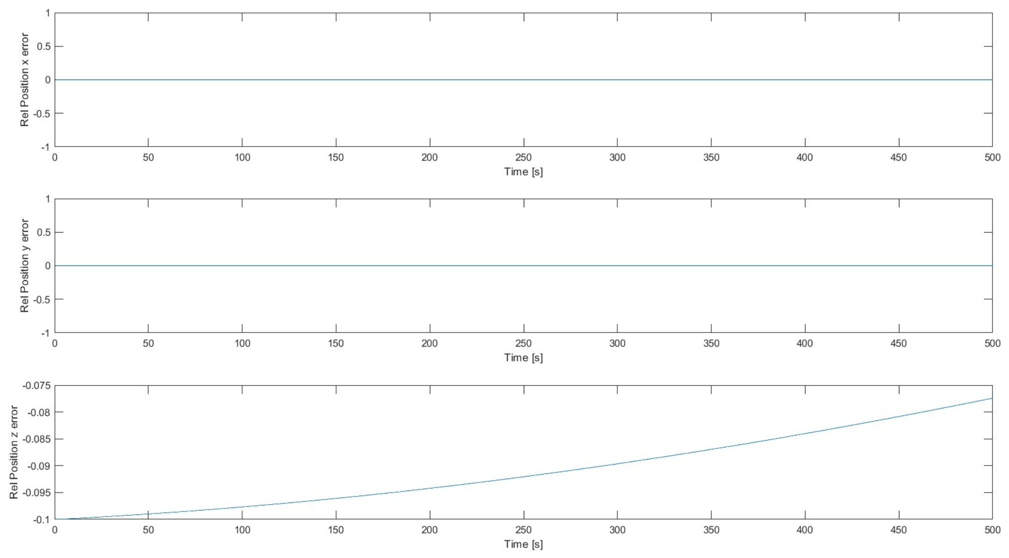


Figure 27: Particle Filter Estimate error for Relative Position

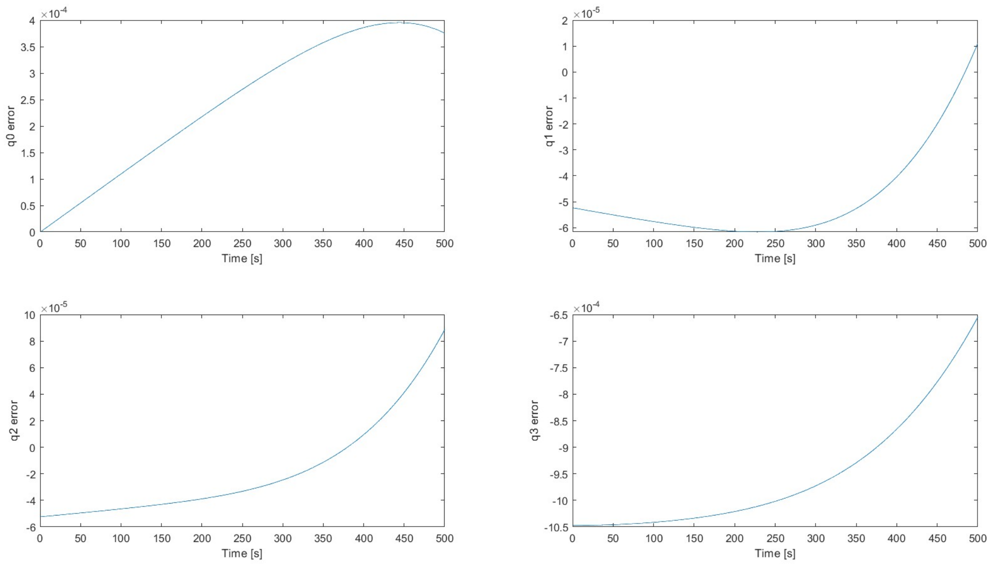


Figure 28: Particle Filter Estimate error for Relative Quaternion

6 | Conclusions and future developments

It is noted that the relative navigation is an important aspect for close proximity operations. Here, we have modelled the propagation of relative motion and used ArUco markers to perform vision navigation. Then particle filter was used to improve those estimates. It can be seen that the particle filter performs well to estimate those parameters.

Many further studies can be done related to this topic. Multiple cameras can be used to perform vision navigation to get better results. The estimation methodology can be changed to other filters or related to deep learning techniques. There are several other fiducial markers available and their use can be tested for spacecraft relative navigation.

Bibliography

[1] Detection of aruco markers. URL https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html.

[2] Blender and opencv camera frame. URL https://stackoverflow.com/questions/64977993/applying-opencv-pose-estimation-to-blender-camera.

[3] Blender software, . URL https://www.blender.org/.

[4] Blender camera matrix, . URL https://blender.stackexchange.com/questions/38009/3x4-camera-matrix-from-blender-camera/120063#120063.

[5] Prisma mission. URL https://www.eoportal.org/satellite-missions/prisma-prototype#gnc-guidance-navigation-and-control-experiments.

[6] Pangu software. URL https://www.star-dundee.com/products/pangu-planet-and-asteroid-natural-scene-generation-ut#technical_specs/.

[7] Camera calibration and 3d reconstruction. URL https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html.

[8] Surrender software. URL https://www.airbus.com/space/space-exploration/SurRenderSoftware.html.

[9] Xss-11 mission. URL https://www.kirtland.af.mil/Portals/52/documents/AFD-111103-035.pdf?ver=2016-06-28-110256-797c.

[10] A. a Better Rendezvous in Space. URL https://www.nasa.gov/mission_pages/station/research/news/b4h-3rd/it-automating-better-space-rendezvous/.

[11] J. P. Alepuz, M. R. Emami, and J. Pomares. Direct image-based visual servoing of free-floating space manipulators. *Aerospace Science and Technology*, 55:1–9, 2016.

[12] K. T. Alfriend, S. R. Vadali, P. Gurfil, J. P. How, and L. Breger. Spacecraft formation flying: Dynamics, control and navigation. 2009.

[13] R. Ambrose, I. Nesnas, and R. Mueller. Nasa technology roadmaps: Ta 4: Robotics and autonomous systems. Technical report, NASA, 2015.

[14] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5): 698–700, 1987. doi: 10.1109/TPAMI.1987.4767965.

[15] P. Bodin, R. Noteborn, R. Larsson, T. Karlsson, S. D'Amico, J. S. Ardaens, M. Delpech, and J.-C. Berges. The prisma formation flying demonstrator: Overview and conclusions from the nominal mission. *Advances in the Astronautical Sciences*, 144:441–460, 2012.

[16] J. A. Christian and S. Cryan. A survey of lidar technology and its use in spacecraft relative navigation. NASA, 2013.

[17] W. H. Clohessy and R. S. Wiltshire. Terminal guidance system for satellite rendezvous. *Journal of the Aerospace Sciences*, 27:653–658, 1960.

[18] H. D. Curtis. Orbital mechanics for engineering students. 2005.

[19] H. Damirchi, R. Khorrambakht, and H. Taghirad. Aras-iref: An open-source low-cost framework for pose estimation. pages 303–308, 11 2019. doi: 10.1109/ICRoM48714.2019.9071852.

[20] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. 2008.

[21] L. Ferraz, X. Binefa, and F. Moreno-Noguer. Very fast solution to the pnp problem with algebraic outlier rejection. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–508, 2014.

[22] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `https://doi.org/10.1145/358669.358692`.

[23] N. J. Gordon, D. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. 1993.

[24] F. Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010. doi: 10.1109/MAES.2010.5546308.

[25] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. *2011 International Conference on Computer Vision*, pages 383–390, 2011.

[26] R. Holst. Satellite attitude control using magnetorquers with magnetic dipole moment cancellation. Master's thesis, Aalborg University, 2014.

[27] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of The Optical Society of America A-optics Image Science and Vision*, 5:1127–1135, 1988.

[28] A. S. Komanduri. *Guidance and Control of a Spacecraft to Rendevous and Dock with a Non-cooperative Target.* Cuvillier, 2011.

[29] R. Kroes, O. Montenbruck, W. Bertiger, and P. Vissern. Precise grace baseline determination using gps. *GPS Solutions*, 9:21–31, 2005.

[30] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81:155–166, 2008.

[31] G. D. Mauro, M. Lawn, and R. Bevilacqua. Survey on guidance navigation and control requirements for spacecraft formation-flying missions. *Journal of Guidance, Control, and Dynamics*, pages 1–22, 2017.

[32] C. J. Montalvo. *Aerospace Mechanics and Controls.* 2022.

[33] R. Opromolla, G. Fasano, G. Rufino, and M. Grassi. A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. *Progress in Aerospace Sciences*, 93:53–72, 2017. ISSN 0376-0421. doi: https://doi.org/10.1016/j.paerosci.2017.07.001.

[34] L. Patrioli. Development of a localization system based on aruco markers for a small space platforms test bench uncooperative space objects. Master's thesis, Politecnico di Torino, 2020.

[35] V. Pesce. Stereovision-based pose and inertia estimation for unknown and uncooperative space objects. Master's thesis, Politecnico di Milano, 2014.

[36] V. Pesce. *Autonomous Navigation for Close Proximity Operations around Uncooperative Space Objects.* PhD thesis, Politecnico di Milano, 2018.

[37] D. Pinard, S. Reynaud, P. Delpy, and S. E. Strandmoe. Accurate and autonomous navigation for the atv. *Aerospace Science and Technology*, 11:490–498, 2007.

[38] M. Pittoni. Monocular feature-based navigation in proximity of unknown space ob-

jects with regions of interest effective extraction. Master's thesis, Politecnico di Milano, 2020.

[39] A. Rathinam and Y. Gao. On-orbit relative navigation near a known target using monocular vision and convolutional neural networks for pose estimation. i-SAIRAS, 2020.

[40] M. D. Shuster and S. D. Oh. Three-axis attitude determination from vector observations. *Journal of Guidance Control and Dynamics*, 4:70–77, 1981.

[41] J. A. Starek, B. Açıkmese, I. A. Nesnas, and M. Pavone. Spacecraft autonomy challenges for next-generation space missions. *Advances in Control System Technology for Aerospace Applications*, pages 1–48, 2016.

[42] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987. doi: 10.1109/JRA.1987.1087109.

[43] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:376–380, 1991.

[44] D. A. Vallado. Fundamentals of astrodynamics and applications. 1997.

[45] X. Wang, W. Qin, Y. Bai, and N. Cui. A novel decentralized relative navigation algorithm for spacecraft formation flying. *Aerospace Science and Technology*, 48: 28–36, 2016.

[46] L. Zhang, S. Zhang, H. Yang, H. Cai, and S. Qian. Relative attitude and position estimation for a tumbling spacecraft. *Aerospace Science and Technology*, 42:97–105, 2015.

A | Appendix A

A.1. Calibration Matrix from Blender

The following code is used to obtain the camera intrinsic matrix from Blender[4].

```
scene = bpy.context.scene
f_in_mm = camd.lens
scale = scene.render.resolution_percentage / 100
resolution_x_in_px = scale * scene.render.resolution_x
resolution_y_in_px = scale * scene.render.resolution_y
sensor_size_in_mm =camd.sensor_width
pixel_aspect_ratio = scene.render.pixel_aspect_y / scene.render.pixel_aspect_x
view_fac_in_px = resolution_x_in_px
pixel_size_mm_per_px = sensor_size_in_mm / f_in_mm / view_fac_in_px
s_u = 1 / pixel_size_mm_per_px
s_v = 1 / pixel_size_mm_per_px / pixel_aspect_ratio
u_0 = resolution_x_in_px / 2 - camd.shift_x * view_fac_in_px
v_0 = resolution_y_in_px / 2 + camd.shift_y * view_fac_in_px / pixel_aspect_ratio
skew = 0 # only use rectangular pixels
K = Matrix(
...         ((s_u, skew, u_0),
...         (   0,  s_v, v_0),
...         (   0,    0,   1)))
>>> K
Matrix(((7111.111328125, 0.0, 320.0),
        (0.0, 7111.111328125, 240.0),
        (0.0, 0.0, 1.0)))
```

A.2. Code to interface MatLab output to Blender

```
import bpy
import scipy.io
import numpy
import os
import glob
files=glob.glob('/Users/Jim/Desktop/*')
for f in files:
    os.remove(f)
mat = scipy.io.loadmat('relativevaluestrue.mat');
data=mat['blenderout'];
Target_X=data[:,0];
Target_Y=data[:,1];
Target_Z=data[:,2];
Eulerx=data[:,3];
Eulery=data[:,4];
Eulerz=data[:,5];
rotation_steps=len(Eulerx);

def rotate_and_render(output_dir, output_file_pattern_string = 'render%d.jpg',
rotation_steps=len(Eulerx), cube =bpy.context.editable_objects[0]):
  import os
  import math
  cube.rotation_mode='XYZ';
  original_rotation = cube.rotation_euler
  original_location=cube.location
  for step in range(0, rotation_steps):
    cube.location.x=Target_X[step];
    cube.location.y=Target_Y[step];
    cube.location.z=Target_Z[step];
    cube.rotation_euler.x = math.degrees(Eulerx[step]);
    cube.rotation_euler.y = math.degrees(Eulery[step]);
    cube.rotation_euler.z = math.degrees(Eulerz[step]);
    bpy.context.scene.render.filepath = os.path.join(output_dir,
    (output_file_pattern_string % step))
    bpy.ops.render.render(write_still = True)
  cube.rotation_euler = original_rotation
  cube.location=original_location
rotate_and_render('/Users/Jim/Desktop/Measurement_data', 'render%d.jpg')
```

Acknowledgements

I would like thank my advisor Dr. Mauro Massari for giving me the opportunity to work on this topic and for his guidance and support throughout this project. I would also like to thank Toon Stolk and Deepana Gandhi who provided valuable comments and suggestions at certain critical parts of the project. I would also like to thank my friends and family for being patient and for their continuous support and understanding when undertaking this degree. Finally, I would like to thank God for providing me the strength and invaluable guidance when faced with difficulties and to be able to complete my degree.