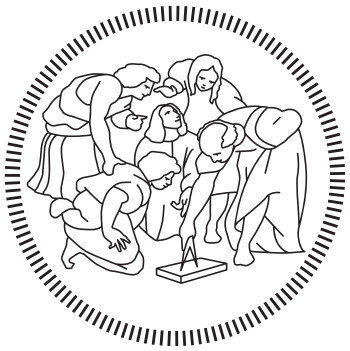
Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Spaziale



Analysis and validation of spaceborne synthetic imagery using a Vision-Based pose initialization algorithm for non-cooperative spacecrafts

Advisor: Prof. Paolo LUNGHI
Co-Advisor: Dr. Eng. Aureliano RIVOLTA

Thesis by:
Francescodario CUZZOCREA Matr. 885016

Academic Year 2019–2020

A chi mi vuole e mi ha sempre supportato...

Ringraziamenti

Statement of original authorship

I, Francescodario Cuzzocrea, confirm that this Master's Thesis has been written solely by the undersigned and contains the work of no other person or persons except where explicitly identified to the contrary. I also state that said Master's Thesis has not been submitted elsewhere for the fulfilment of any other qualification. I make this statement in full knowledge of and understanding that, should it be found to be false, I will not receive a grade and may face disciplinary proceedings.

Francescodario Cuzzocrea, April 1, 2021, Milano

Abstract

The capability of doing close range proximity operations between spacecrafts to either repair, refuel or deorbit end of life and nonfunctional is more and more becoming a key-enabling factor for future space missions. When approaching uncooperative spacecrafts (no markers or other specific supportive means) it is also required that the spacecraft which performing the approach must be able to perform onboard estimates the pose (*i.e.*, relative position and attitude) in order to correctly move towards - and eventually grasp - the target space object. The usage of vision-based sensors for pose estimation is particularly attractive to solve the problem of pose determination due to their low volumetric and power requirements, particularly in comparison to other systems such as LIDaR. The capability of rendering thousands of images of the the target space object is particularly important when developing vision-based techniques aiming at solving the pose problem. For example, deep learning techniques relies on large annotated data-sets of images. For what concerns terrestrial applications, there are a plethora of data-sets commonly available, however for spaceborne applications there is a general lack of such data-sets. The main reason arises from the difficulty of acquiring realistic images of the desired space object which can be considered representative of a data-set taken by a real monocular camera. The purpose of this thesis is to try to to overcome this limitation by presenting a method based on ray tracing which let the user generate thousand of images of a desired spacecraft given its CAD model and the desired attitude, which will be then used to implement and test a vision-based pose estimation algorithm. To reach the aforesaid purposes, the first thing made is to identify how to correctly model the scene which is being represented, by identifying all the correct optical parameters to assign to the various Earth's surfaces (such as oceans, clouds and terrains) and to the spacecraft's surfaces (which can be made of composite materials, metallic materials, covered with MLI or solar panels) which are being represented. Next, is important to set the correct illumination conditions (so, simulate sunlight). To have an image as much realistic as possible, it is of crucial importance also to simulate the camera behavior in terms of aperture angle and intrinsic noise of the sensor. Once the images are generated, they can be analyzed by using the desired vision-based algorithm. For what concerns this work, the images have been used to analyze and validate an innovative algorithm meant to solve the pose

initialization problem called "SVD algorithm". The "SVD algorithm" tries to estimate the unknown pose of the target by matching the geometrical information about the target (in the form of its 3D CAD model), typically stored on-board the spececraft which is performing the servicing maneuvers, with 2D geometrical information extracted from the image using edge detection techniques. The "SVD algorithm" adds some improvements with respect to the current state of the art of edge detection techniques, in particular it offer: the weak gradient eliminator technique to select a region of interest around the target spacecraft in the image, the usage of parameters which can be scaled with distance for what concerns the edge detection procedure and a smarter organization of the features extracted from the 2D images, which stores them in "perceptual groups", which allows to obtain a sensible reduction of the matching combinations. The thesis ends by comparing the generated data-set of images against the SPEED data-set - the first publicly available machine learning set of synthetic and real spacecraft imageries - and with a preliminar validation of the SVD algorithm.

Sommario

La possibilità di effetuale manovre a corto raggio in contesti spaziali al fine di compiere missioni di riparazione, rifornimento, rimozione di satelliti non più operativi o detriti spaziali privi di controllo d'assetto sta assumendo sempre più importanza per la programmazione delle future missioni spaziali. Nelle missioni che prevedono l'avvicinamento ad un satellite non-cooperante (cioè non avente alcun tipo di marcatore o supporto che permetta di conoscere a priori il suo assetto relativo), è altresì molto importante che il satellite che effettua le operazioni di avvicinamento sia anche in grado di calcolare con sufficiente precisione l'assetto e la distanza relativa rispetto al *target*, in modo da potersi avvicinare correttamente ed eventualmente afferrare il satellite *target*. In questo contesto, l'impiego di sensori di tipo *vision-based* per il calcolo di assetto e posizioni relativi risulta particolarmente interessante per via delle ridotte dimensioni e del ridotto assorbimento di potenza che caratterizza i sensori di tipo ottico, come può essere una telecamera, rispetto ad altre soluzioni come per esempio quelle basate su sensori LIDaR. Per poter implementare tecniche di controllo e navigazione *vision-based*, la posibilità di renderizzare migliaia di immagini dello scenario operativo risulta dunque una tecnologia chiave. Per esempio, le tecniche di *deep learning* si basano sull'analisi di grandi *data-set* di immagini. Per quanto riguarda le comuni applicazioni terrestri, la disponibilità di un data-set non rappresenta oggigiorno un problema, data la pletora di immagini disponibili. Lo stesso tuttavia non si può dire per quanto riguarda invece le applicazioni spaziali. Le principali ragioni sono rappresentate dalla ogettiva difficoltà di acquisire immagini sufficientemente realistiche e che possano considerarsi come rappresentative di un *data-set*. Lo scopo di questa tesi è dunque quello di cercare di superare questa limitazione impiegando tecnologie di *ray tracing* al fine di creare uno strumento in grado di produrre automaticamente migliai di immagini di un determinato satellite, noto il suo modello CAD e l'assetto desiderato. Le immagini così prodotto verranno poi utilizzate al fine di implementare e testare un algoritmo *vision-based* atto a stimare assetto e posizione relativi del satellite *target* rappresentato nelle immagini. Per il raggiungimento dei suddetti obiettivi, la prima cosa da fare è capire come modellare correttamente la scena da rappresentare, andando ad individuare quali sono i corretti parametri ottici da assegnare alle diverse superfici terrestri (oceani, nuvole e terreno) ed alle superfici del satellite (superfici che possono essere di materiale composito, metallico, rivestite

da MLI o pannelli solari) che vogliamo rappresentare. Successivamente, è necessario impostare le corrette condizioni di luminosità (ovvero, simulare correttamente la luce solare). Al fine di avere un immagine quanto più realistica possibile è altresì importante simulare correttamente anche il comportamento della camera, in termini di angolo di apertura, risoluzione e rumore intrinseco del sensore. Una volta prodotte le immagini, esse possono essere analizzate utilizzando la metodologia desiderata. Per quanto riguarda questo lavoro, si è scelto di implementare per un algoritmo innovativo per il calcolo di assetto e posizione relativi di un satellite *target* non cooperante, denominato "algoritmo SVD". L'algoritmo SVD cerca di stimare assetto e posizione relativi del satellite *target* andando ad effettuare un *match* fra le informazioni geometriche del *target* stesso (sotto forma di modello CAD tridimensionale) - tipicamente presenti all'interno del satellite che effettua l'avvicinamento - con features bidimensionali estratte dall'immagine (sotto forma di segmenti appartenti alla *silhouette* del satellite *target*) tramite tecniche di *edge detection*. L'algoritmo SVD apporta alcuni miglioramenti rispetto a quello che è lo stato dell'arte, in particolare presenta: una tecnica di eliminazione dei gradienti deboli per determinare una regione di interesse che definisca dove si trovi il satellite *target* all'interno dell'imagine, la definizione di parametri scalabili sulla distanza per quanto riguarda la procedura di *edge detection* ed un processo di catalogazione delle *features* che le classifica in gruppi geometrici, il quale consente di ottenere una consistente riduzione delle possibilità di matching combinando intelligentemente i suddetti gruppi geometrici. La tesi si conclude quindi proponendo al lettore una comparazione qualitativa fra le immagini sintetiche sviluppate per questo progetto e le immagini appartenenti al data-set SPEED - il primo *data-set* comprendente immagini sintetiche e reali disponibile pubblicamente - e con una validazione preliminare dei risultati ottenuti applicando l'algoritmo SVD sulle prime.

Host Company

This R&D project has been developed at D-Orbit. D-Orbit is a New Space company with solutions covering the entire life-cycle of a space mission, including mission analysis and design, engineering, manufacturing, integration, testing, launch, mission control and end of life decommissioning. The company's competitive advantage is in the versatility of its launch and deployement services that can me tailored to the customer's needs, from the launch procurement of a single satellite using standard deployment strategies to the precise deployment of a full constellation with ION satellite Carrier, a satellite dispenser developed and operated by D-Orbit. The ION Satellite Carrier can host any combination of CubeSat with a total volume of up to 48U and release them individually into distinct orbital slots, enabling deployement schemes previously unavailable to satellite with no independent propulsion. Committed to pursuing business model that are profitable, friendly for the environment, and socially beneficial, D-Orbit is the first certified B-Corp space company in the world.
Headquartered in Como, Italy, D-Orbit has subsidiaries in Lisbon, Portugal, Harwell, UK, and Washington DC, USA.

Contents

List of Figures

List of Tables

List of Symbols

α Aperture angle

λ_{Hough} Maximum gap between two points to be considered on the same line segment

$A.R.$ Aspect Ratio

$\mathbf{A_{CN}}$ Orientation of the camera frame with respect to the inertial frame

$\mathbf{A_{TC}}$ Orientation of target principal axes with respect to the camera frame

$\mathbf{A_{TN}}$ Orientation of the target principal axes with respect to the inertial frame

L_C S/C characteristic length

$L_{min,Hough}$ Expected minimum lenght of a line segment

N_u Number of horizontal pixels

N_v Number of vertical pixels

d_u Horizontal pixel length

d_v Vertical pixel length

f_x Orizontal focal length

f_y Vertical focal length

$\mathbf{t_C}$ Target center of mass location with respect to camera frame

Abbreviated Terms

2-D Two-Dimensional

3-D Three-Dimensional

ADR Active Debris Removal

CCD Charge Couple Device

CDF Cumulative Distribution Function

CG Center of Mass

CV Computer-Vision

DCM Direction Cosine Matrix

ECIF Earth Centered Inertial Frame

EO Electro-Optical

FF Formation Fliying

LIDaR Light Detection and Ranging

LoC Lines of Code

NR Netwon-Raphson

OOS On-Orbit Servicing

P-n-P Perspective-n-Point

PDF Probability Distribution Function

POV-Ray The Persistence Of Vision Raytracer

R&D Research and Development

ROI Region of Interest

S&H Sobel and Hough

S/C Spacecraft

SDL Scene Description Language

STL Stereolitographic

SVD Sharma-Ventura-D'Amico

TM Template Matching

WGE Weak Gradient Eliminator

Introduction

"All models are wrong, but some are useful."

George Box

Motivation

Close range proximity operations between Spacecraft (S/C)s has been studied and discussed by space agencies and private companies since the early stages of space exploration, dating back to the Apollo program [37]. Since then we can find a wide range of missions where close-range proximity operations are in, like Formation Fliying (FF) [5] [2], On-Orbit Servicing (OOS) [55] [67] [62] [1] and Active Debris Removal (ADR) [11] [7]. Most of those missions were possible thanks to the presence of on-board crew or to the cooperativeness between S/Cs. In a typical scenario, the S/C which is being serviced is referred as "target" or "client", while the S/C which is performing the approach is referred as "chaser" or "servicer". A target is deemed cooperative if it is built to provide information suitable for the estimation of its distance and orientation in space with respect to the chaser S/C. Also, it can be be actively or passively cooperative depending on whether it interacts with a dedicated radio-link with the chaser S/C or not [47]. As regard to the new generation of space robotics missions such as debris removal and OOS, proximity operations and docking are key-enabling capabilities for either repair, refuel or deorbit end-of-life and nonfunctional S/Cs [64]. The main challenge when performing close-range operations in actual OOS and ADR missions is when the target S/C may be uncooperative. This implies that the target S/C may not be equipped with an active communication link or identifiable markers such as light-emitting diodes or corner cube reflectors to help with computing the relative position and attitude of the active S/C (chaser) with respect to a uncooperative target space object [57]. Another important aspect, which comes out when dealing with uncooperative targets, is that debris or operating S/Cs to be serviced may have suffered physical damages as well as optical degradation of their surfaces due to the prolonged exposure to the space environment, thus appearing different

than expected [47]. Thus, when operating in close-proximity the attitude and the motion of the target S/C must be estimated in autonomy by exploiting the sensors available on the S/C actively performing the servicing maneuvers. With regards to the technological aspects, Electro-Optical (EO) sensors have been identified as the best option for relative navigation in the foredescribed scenario [47] [50]. Either active Light Detection and Ranging (LIDAR) systems or passive monocular and stereo cameras can be used. The selection of the navigation sensor must consider the resources available on board in terms of mass, electrical and processing power, on one side, the mission scenario and the costs to be sustained for design and development of the satellite segment, on the other side [11] [50]. As stated in [58] monocular vision navigation has been identified as an enabling technology for present and future FF and OOS missions (namely PROBA-3 by ESA [10], PRISMA by OHB Sweden [19]). Monocular navigation on such missions relies on finding an estimate of the initial pose of the space resident object with respect to the camera, based on a minimum number of features from a 3D computer model and a single 2D image [58]. In contrast to other state-of-the-art systems based on Light Detection and Ranging (LIDaR) or stero camera sensors, monocular navigation ensures rapid pose determination while offering some advantages such as lower hardware complexity, cost, weight and power consumption, possibility to be simultaneously used for supervised applications and a much larger operational range, not limited by the size of the platform [60] [64] [50]. However, the benefit of lower hardware complexity trades off with increased algorithmic complexity since a monocular sensor cannot provide direct three-dimensional (3D) measurements about the target. Moreover, monocular sensors can be less robust to adverse illumination conditions typical of the space environment [65] (e.g., saturation under direct Sun illumination, or absence of light during eclipse) [50]. The increasing challenges of space exploration and moreover the urgent need for debris removal to free slots in orbit and to avoid unwanted collisions is what mainly motivate this work. The capability of being able to develop and test pose determination algorithm in fact will be a key factor for the overall success of new generation missions. Thus, this work is motivated from the need to implement a low-cost, reliable and extendable solution for simulating low Earth orbit spaceborne imagery for Computer-Vision (CV) algorithms needs.

Problem Statement

The availability of a tool capable of rendering image of the target client S/C is of vital importance for the implementation and testing of any kind of CV algorithm. The usage of artificial in images in fact gives a complete control over the scene and more importantly, recently investigated deep learning techniques relies on very huge annotated data-sets of images. The main difficulties when searching for suitable data-sets arises from the difficulty of acquiring thousands of spaceborne

images of the desired target S/C with accurately annotated pose labels and with an high enough degree of realism, capable of being representative of real operative conditions. A poorly made data-set in fact can lead to incoherent results due to wrong assumptions, especially when developing techniques which are based on the implementation algorithms relying on Neural Networks. A standard tool capable of generating thousands of images at demands moreover enable the user to systematically evaluate and compare the performance of different algorithms. So, in this work the possibility of generating an accurate enough dataset by employing Ray-Tracing techniques will be firstly explored. After that, the generated data-set will be employed to solve the relative pose estimation problem. As stated in [16] and [60], the pose estimation problem consist of determining the position of the relative position and the relative attitude of target space object with respect to an active S/C which act as a servicer. Several different technological solutions can be employed to archive the goal, according to the mission budget, the mission scenario and the cooperativeness of the target. This work will be focused on the determination of the unknown 3-D pose a non-cooperative target (no markers or other specific supportive means) using a single Two-Dimensional (2-D) image, given the target's Three-Dimensional (3-D) geometrical representation (referred as map or model).

Structure of the Thesis

The thesis is divided into four chapters. The first chapter presents to the reader a brief overview on the current state-of-art status for what concerns synthetic images generation for spaceborne applications as well as the autonomous close-proximity relative navigation problem. The second chapter is completely focused on the image generation problem, with an in-dept explanation regarding the toolbox developed to allow the end user to produce image data-sets starting from the CAD model of the target S/C. The third chapter is dedicated to the analysis and the implementation of the SVD architecture for solving the pose initialization problem when considering non cooperative S/C. The fourth chapter illustrates the results obtained when using the pose initialization algorithm on the generated images. The thesis ends with a depiction of the possible future developments that can be performed on both the synthetic images and the algorithm.

Chapter 1

State-of-the-art and Limitations

"We are just an advanced breed of monkeys on a minor planet of a very average star. But we can understand the Universe. That makes us something very special."

Stephen Hawking

In this introductory chapter the reader will first be presented with a brief review of the current methods used for generating synthetic images suitable for training CV algorithms. After, a section will be dedicated to illustrate the current state-of-art close-proximity relative navigation techniques which can be employed when using a monocular camera.

1.1 Synthetic Image Generation for Spaceborne Applications

The availability of a tool capable of rendering images of a target S/C is of vital importance for the implementation and testing of any kind of CV algorithm. The use of artificial in images in fact gives a complete control over the scene. As stated in [40], the generated data-set should be as complete as possible in terms of metals and terrain features and illumination conditions. A lack of realism in image generation can lead to incoherent results, not representative of real operative conditions and thus can lead to wrong assumptions and so, wrong results in terms of CV algorithm tuning. In the following sections will follow a brief review of the current tool offered by the industry and of the one used in the academic field to produce synthetic spaceborne imagery.

1.1.1 Commercial Solutions

For professional solutions are intend all those software which can be bought from a private company and so which comes with a license. The most common

professional software which can be used for generating spaceborne synthetic imagery are PANGU developed by ESA and SurRender, developed by Airbus.

ESA PANGU

PANGU is a software developed in order to create synthetic planetary surface images, as much representative as possible, to aid the development of vision-based algorithms. PANGU is a user friendly software which is ready to use. PANGU can also be integrated with proprietary or OSS simulation tools and it gives the possibility to correctly simulate a space camera in all its aspects (so focal lengths, and other relevant parameters of the image). It renders the images using OpenGL and it can use GPU cores to accelerate the rendering. Further information on PANGU can be found in [49].

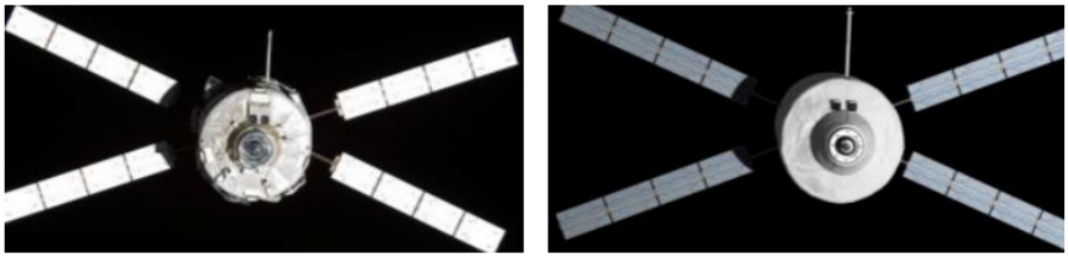


Figure 1.1: Images rendered using PANGU. Comparison between a real (left) and a PANGU (right) image of ESA's ATV spacecraf (taken from the PANGU official flyer).

Airbus Surrender

SurRender is a software developed by Airbus Defense and Space. The software handles various space objects such as planets, asteroids, satellites and spacecraft. It is capable of accommodating solar system-sized scenes without precision loss, and optimizes the ray tracing process to explicitly target objects. It can operate in real time mode to be coupled with proprietary or open source simulation tools and it gives the possibility to have an hardware in the loop simulation to test the responsiveness of the image processing subsystem. It gives the possibility to correctly simulate a space camera in all its aspects (so focal lengths, and other relevant parameters of the image). Parallelization is implemented, so can be ran on cloud platform to accelerate rendering times. Further information regarding the SurRender software can be retrieved in [8].

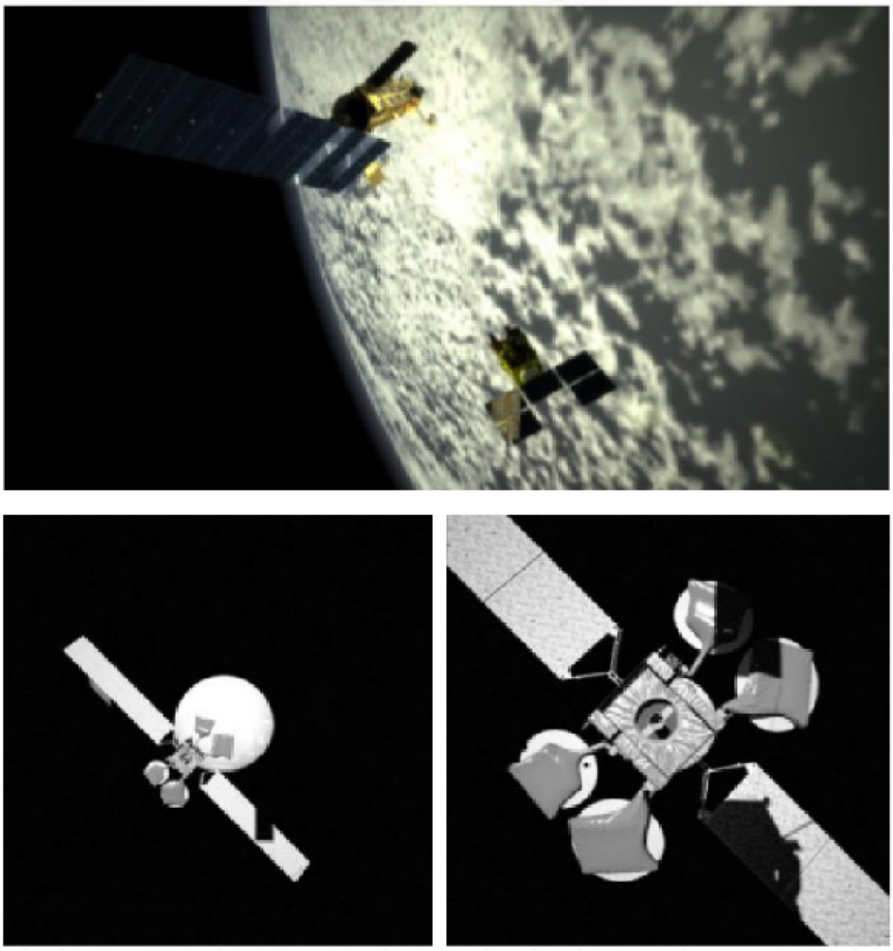


Figure 1.2: Images rendered trough SurRender. Top Panel: The SPOT 5 and ENVISAT satellites rendered with SurRender on an Earth background. Bottom panel: simulation of a rendezvous scenario. A telecom satellite is approached by the SpaceTug. Note the shadow of the tug on the satellite. On the right panel, the tug light spot enlightens the sun shadow [8].

1.1.2 Academic Solutions

For academic solutions are intended all those data-sets which have been generated in order to validate novel architecture to perform pose initialization or, more in general, to develop CV algorithms for space applications, either based on a more traditional edge detection approach or on the usage of neural networks. The two most prominent data-set available at the moment of writing this thesis are the SPEED data-set, developed at Stanford University, and the URSO data-set developed at University of Surrey.

SPEED dataset: image generation using OpenGL

The speed data-set represents the first publicly free available machine learning data-set for S/C pose estimation [36]. It consist in 15000 synthetic and 200 real images of the Tango S/C from the PRISMA mission. Specifically, the real images have been captured using the TRON facility of the Space Rendezvous Laboratory, at Stanford University, while the synthetic images have been generated simulating the internal camera properties of a Point Grey Grasshopper 3 camera with a Xenoplan 1.9/17 mm lens [59],[57]. The synthetic images of the Tango S/C are created using the camera emulation software of the Optical Stimulator [6], which uses an OpenGL based image rendering pipeline to generate photo-realistic images of the Tango S/C with an imposed ground-truth pose. Half of the generated synthetic images presents random real images of the Earth captured by the Himawari-8 geostationary weather satellite. Illumination conditions for those images are set up in order to match the background Earth images. All images are then processed with Gaussian blurring and zero-mean. The interested reader can find more details about the SPEED data-set in [36] and [6].

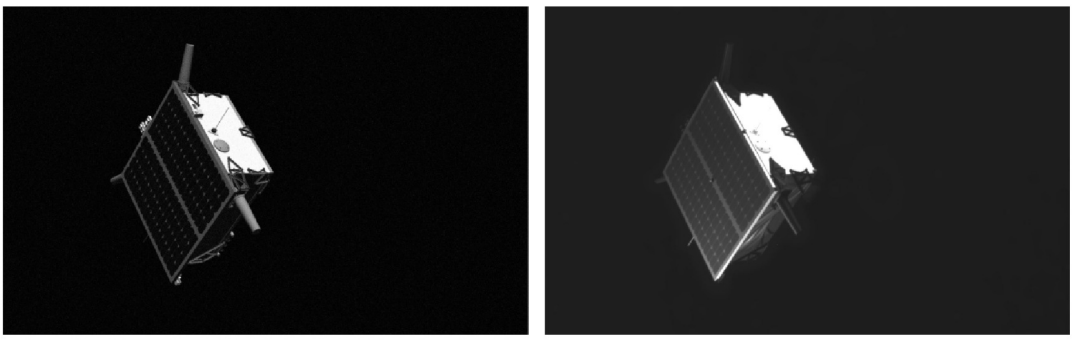


Figure 1.3: Left: SPEED synthetic imagery. Right: SPEED real imagery [36].

URSO data-set: image generation using Unreal Engine 4

The URSO (Unreal Rendered Spacecraft On-Orbit) data-set is a synthetic image data-sets of Soyuz and Dragon spacecraft models orbiting the Earth, rendered

using a custom simulator based on Unreal Engine 4, which is a rendering engine usually emplyed in the video-games field. Images belonging to the URSO data-set were rendered at a resolution of 1080×960 pixels by a virtual camera with a $90°$ horizontal field-of-view and auto-exposure. Authors of URSO points out state-of-the-art game engines, such as the Unreal Engine 4 are widely used for the training of autonomous driving algorithm [21] and robotics ([56], [42]), however these have also been criticized for the lack of photometric accuracy of the camera sensors [8]. Despite that, authors says that recent efforts have been made in the source-available UE4 to implement physically-based shading models and cameras and claims that their custom simulator, called URSO, allows obtaining photorealistic images and depth masks of commonly used spacecrafts orbiting the Earth. Further information regarding the URSO data-set can be found in [53].

Figure 1.4: Example of frames synthesized by URSO of a soyuz mode [53].

1.2 Monocular Based Pose Estimation

An accurate and trustworthy solution to the pose estimation problem represents a key element of any navigation system and it's of crucial importance for the future advancement of ADR, FF and OOS missions. The target cooperativeness plays a key role in the selection of the sensor suite of the chaser S/C as well as in the implementation of the pose solver algorithms. The problem of determining the pose of a non-cooperative target utilizing an image captured my a monocular sensor and the information gathered by the knowledge of its 3-D model has been extensively addressed in the field of CV, mostly regarding terrestrial application. Dhome proposed a closed-form solution to solve for the pose given the correspondences between the edges detected in the image and the line segments of the 3D model [20]. The pose was computed using of an exhaustive predictor-corrector method which matched all the possible set of the 3-D model line segments with the 2-D edges detected in the image. Eventually, to cope with the issue of doing an exhaustive search for correspondences soft-assign has been used [27], [17], however this has been demonstrated to be slower with respect to the Random Sample Consensus (RANSAC) method in [4]. Numerous other algorithms for finding the object's pose from the image and model feature correspondences have been proposed [43], [66], [38], however all those algorithms are able to produce a correct pose solution only if several correct feature correspondences are provided, which limits the real-time application of these algorithms to the pose initialization problem. The increasing challenges with regards the space exploration, such as performing FF or OOS missions and more importantly, the urgent need to perform ADR missions has brought increased interest to the importance of analyzing the pose initialization problem also for spacecraft applications. When handling non-cooperative targets usually the process of pose determination is carried out in two step, the first one is the acquisition phase, while the second one is the tracking phase. The acquisition phase is carried out when the first batch of images is acquired by the camera and no *a priori* knowledge is present about the pose of the target S/C. The tracking phase instead starts as soon as new images are acquired by the camera. The new pose of the target is computed, also using the knowledge of the estimates at one or more previous time instants. Optionally, a pose refinement step can be implemented in order to improve the accuracy of the pose estimate. Under the assumption of known target (so, basic information about the target geometry are available), the uncooperative pose determination is usually archived by using model-based algorithms. When dealing with monocular sensors, the classical approach involves estimating the unknown pose of the target S/C by matching geometrical information about the target S/C stored on-board (for example a wireframe CAD model) with handcrafted features from the target S/C from the 2-D image. This approach is called feature-based approach. When monocular sensors are used, feature-based algorithms can rely either on P-n-P solvers or Template Matching (TM) techniques. The Goddard Natural Feature

Image Recognition (GNFIR) for example, is an edge tracking algorithm which is based on the usage of the Sobel edge detector [47]. Specifically, once the edge image is determined and given an initial estimate of the pose parameters, the target S/C pose is estimated exploiting the approach proposed in [22]. In absence of the initial estimate, GNFIR can run in acquisition mode and autonomously generate an initialization by using a prediction of the motion of both the chaser and the target. However, in orbit tests demonstrated that generating the initial estimate of the pose was an issue for GNFIR, and tracking operations were not accomplished correctly for proximity scenarios in which the target/chaser relative dynamics was fast [16], [45]. In [28] is proposed an approach which relies on the usage of the Harris corner detector [30] for performing the feature detection phase coupled with a linear eight point algorithm to solve for the pose. However, the scheme only provides relative distance information based on background subtraction and Gaussian blob detection algorithms. Testing on actual space imagery has shown that this approach is capable of correctly determining the ROI in the image. TM based techniques are based on the matching of an assigned template with specific features and/or image sections detected in the acquired data. So, TM techniques highly rely on the generation of a relatively large database of template-image of the target in different positions and from different point of views. On this basis, a correlation function is computed in order to evaluate the degree of similarity between each template and the acquired image [47]. Obviously, the unknown pose solution will be provided by the best-correlation template. As reported in [47], Astrium Satellites proposed a fully monocular-based pose determination architecture in the framework of the Debritor program. A silhouette TM algorithm is applied to a two degrees-of-freedom database which is built off-line and organized as a hierarchical view graph [54], thus estimating two relative attitude parameters. Then, a particle filtering framework, initialized by means of an object segmentation procedure, is used to estimate the remaining unknowns, which are the horizontal and vertical offsets, the scale factor and the rotation around sensor boresight [51]. The pose tracking is then performed by applying the edge tracking method described in [14], modified to improve robustness to outliers. As noted in [57] however, in the absence of an extensive database of pre-computed renderings of the target, approaches based on TM tends to be very limiting as small changes in the target orientation and position may significantly affect its appearance, thus a particular care should be taken when dealing with them. The ULTOR Passive Pose and Position Engine (ULTOR P3E) by Advanced Optical Systems, Inc (AOS) is an algorithm developed and tested ([44], [45]) to perform six degrees-of-freedom pose determination of an uncooperative spacecraft by processing monocular images. ULTOR uses simulated imagery of the target to build a database offline that is used to identify natural geometric structures of the target surface by means of correlation. After that, relative position and pose data are evaluated by solving the P-n-P problem using a set of 2-D/3-D point correspondences, however, in orbit tests demonstrated that generating the initial estimate of the pose was an issue

[16], [45]. A different approach to solve pose the pose initialization problem of non cooperative target based on monocular images has finally been proposed by D'Amico *et al.* in [16], which exploits the concepts of perceptual organization [39] of the edges detected in the image using the Sobel and Hough algorithms. Perceptual groups, which are defined by combinations of lines and points that satisfy specific geometric constraints, can be used as more robust descriptors than individual natural features and they allow to reduce the risk of ambiguous matching. Pose refinement and tracking are carried out by applying a multidimensional Netwon-Raphson (NR) method and a weighted iterative batch least-squares estimator, respectively. As highlighted in [57] however, this approach demonstrated to have two critical limitations. The first one is that the initial pose is dependent on a computationally expensive iterative view-space discretization, not suitable for on-board real-time execution. The second one is that the image processing lacks robustness to illumination and background conditions. As stated in [47], the pose acquisition is one of the most critical task to accomplish due to the fact that the initial guess has to be searched in the entire six degrees-of-freedom-space, and still present two major unresolved issues:

- the high computational load required, which is particularly critical when dealing with fast target-chaser relative dynamics;
- the necessity to having robustness to illumination and background conditions and against the variability of pose conditions, which is particularly critical for space objects which typically have symmetric or nearly-symmetric shapes, thus being prone to produce ambiguous results.

Chapter 2

Synthetic Image Generation

"In math, you're either right or you're wrong."

Katherine Johnson

The availability of a tool capable of rendering batch of images of the target client S/C is of vital importance for the implementation and testing of any CV algorithm. For example, deep learning techniques relies on large annotated data-sets of images. For what concerns terrestrial applications, there are a plethora of data-sets commonly available, however for spaceborne applications there is a general lack of such data-sets. The main reason arises from the difficulty of acquiring thousands of spaceborne images of the desired target S/C with accurately annotated pose labels. In this chapter, the proposed solution to generate synthetic spaceborne imagery will be described and analyzed.

2.1 Reference Frames

Before going trough the details regarding the image generation toolbox developed during this work, it is important to define which are the reference frames of interest for image generation, which for instance are:

- Earth Centered Inertial Frame (ECIF)
- chaser body-fixed frame
- camera frame
- target model frame
- target body-fixed frame

The ECIF frame has its origin at the center of mass of the Earth. This frame has a linear acceleration because of the Earth's circular orbit about the Sun, which however can be neglected for attitude analysis. The x-axis is aligned with the direction of the vernal equinox, which is the equinox which occurs near the first day of spring (with respect to the North hemisphere). The z-axis is aligned with the celestial North Pole. The y-axis is rotated by 90 degrees East about the celestial equator.

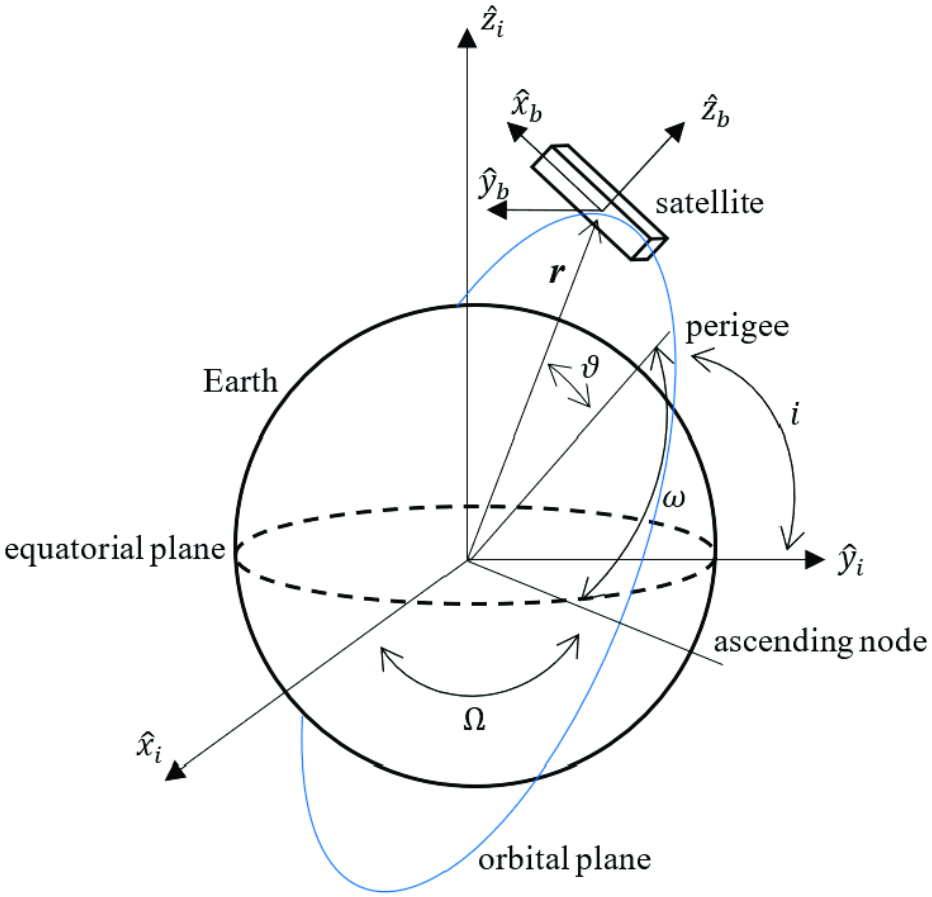


Figure 2.1: Earth Centered Inertial Frame (ECIF)

The chaser and target body-fixed frames have their origins at the CG of both, respectively, and, for the sake of simplicity their axes will be considered to be aligned with their respective principal axes. The origin of the camera frame instead coincides with the center of perspective projection [1]. The target model frame is fixed with respect to the body frame and often coincides with it. For what concerns this work, for the sake of simplicity and without loss of generality, we

[1]The center of projection is the location of the viewer's eye on which projected light rays converge

will consider the camera frame to be coincident with the chaser body-fixed frame, and the target model frame to be coincident with the target body-fixed frame.

2.2 Image Generation

The use of artificial images gives a complete control over the scene. For the purpose of this work, a new procedure for the generation of realistic images, representative of a data-set taken by a monocular camera during a close-proximity approach to target S/C has been developed. Using the illustrated procedure, the user is able to create synthetic images of a target S/C given it's STL model, by fine tuning all the properties of the materials composing the target S/C. Since there could be also cases in which the Earth can be behind the target S/C, the developed tool is also able to simulate Earth's presence at any given location. The tool is also able to the simulate the the atmosphere of the Earth and the cloud layer [29]. The developed tool couples the well known open-source raytracer POV-Ray with MATLAB in order to archive a good degree of realism in the generated images.

2.2.1 Ray Tracing

Ray tracing is a rendering technique used for generating artificial images which relies on the concept of controlling the path of view lines which starts from the observer camera and ends to generic virtual objects and thus calculating the color intensity of the object. What is really of crucial importance for our particular use case is that ray tracing is capable of re-creating some of nature's optical effects through transparent and opaque surfaces as reflection and refraction, scattering, and dispersion phenomena (such as chromatic aberration). Due to this, ray tracing techniques can generate artificial images with a really high degree of photorealism. When the ray tracer renders the scene, a ray of light is traced for every pixel of the camera. Typically, each ray must be tested for intersection with some subset of the objects in the scene. Once the ray has intercepted an object, the ray tracing algorithm will estimate the amount and the type of incoming light at the point of intersection, examine the material properties and by combining those information will calculate the final color intensity that should be attributed to the pixel. Several illumination algorithms and reflective or translucent materials may also require more rays to be re-cast into the scene in order to do the necessary computations. At first glance may seem cumbersome to start the ray from the observer towards the object rather than casting rays to the camera (as is in reality). However, doing so speeds up a lot the computation time, as most of the light rays present in a scene may never reach the eye of the observer, and so, all the time spent for tracing those would be useless. After either a maximum number of reflections or a ray traveling a certain distance without intersection, the ray ceases to travel and the pixel's value is updated.

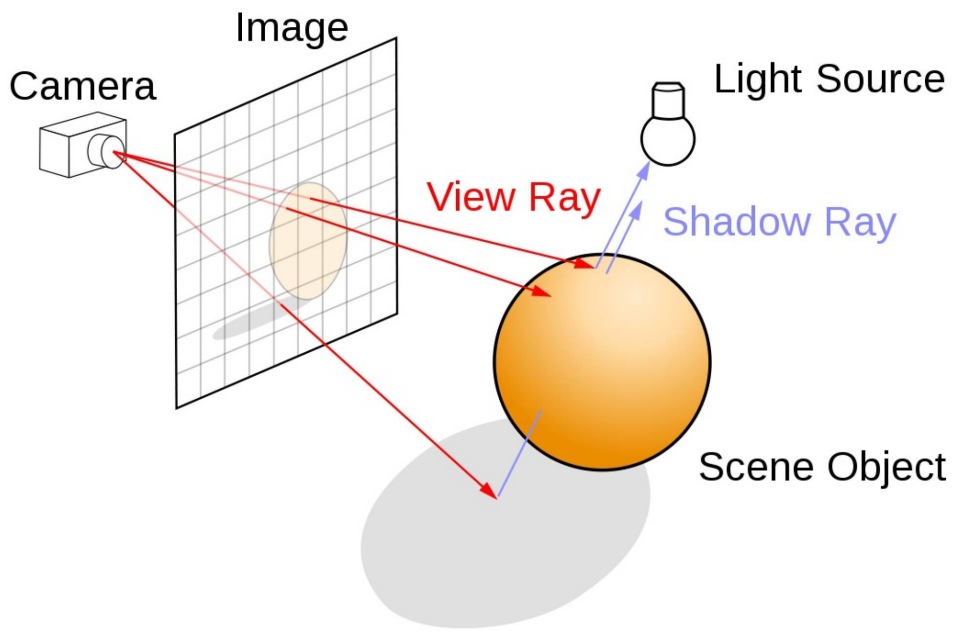


Figure 2.2: Raytracing: from the observer to the light source [13]

For more information regarding how ray tracing works and ray tracing techniques in general, the interested reader can see [35].

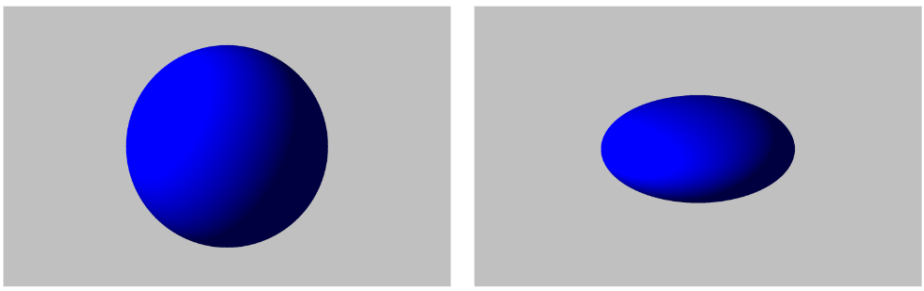
Ray tracing with POV-Ray

POV-Ray is an open-source ray tracing software. It does not offer a GUI for modeling objects, but can be optionally used as Blender rendering engine in order to have a 3-D modeling environment to model objects. POV-Ray has also been already used to generate spaceborne imagery. For example, it has been used under the ESA LunarSim study to render images of lunar surfaces. Being an headless executable, POV-Ray can be easily scripted in order to be used in conjunction with other softwares. POV-Ray gives the programmer several options to customize both the look and feel and the optical properties of the represented objects and the medium that light passes trough. POV-Ray offers the choice to use some predefined geometric shapes, like spheres or cubes; another option instead is to define surfaces of the objects trough meshes. This capability has been used widely by this project to model the spacecraft object. Any surface can then be characterized by its own optical properties. Every object can have a color specified by an Red Green Blue (RGB) triplet or can be wrapped with a texture.
For what concerns light sources, they have no visible shape of their own. They are just points or areas which emit light and can be tuned to accomplish the wanted result. For example we can tune the intensity of the light and the light color by specifying the RGB triplet; any atmospheric effect like opaque gas presence (like smoke or clouds) and its relative optical distortions can be modeled as well.

2.2.2 Environment Modeling

Earth Modeling

Earth modeling has been developed in parallel to another thesis [29]. Here will follow a brief review of how Earth has been modeled taken from [29]. For modeling the Earth, the POV-Ray **sphere** object has been used, in conjunction with the **scale** feature, which allows to make the sphere become an ellipsoid.



(a) Normal sphere (b) Differently scaled sphere

Figure 2.3: Sphere manipulation with POV-Ray [29]

Once the 3-D object is created, the most natural choice one can think of is to wrap a 2-D texture of the Earth (figure 2.4) on it, and this indeed is what has been done, using an high resolution texture of the Earth.

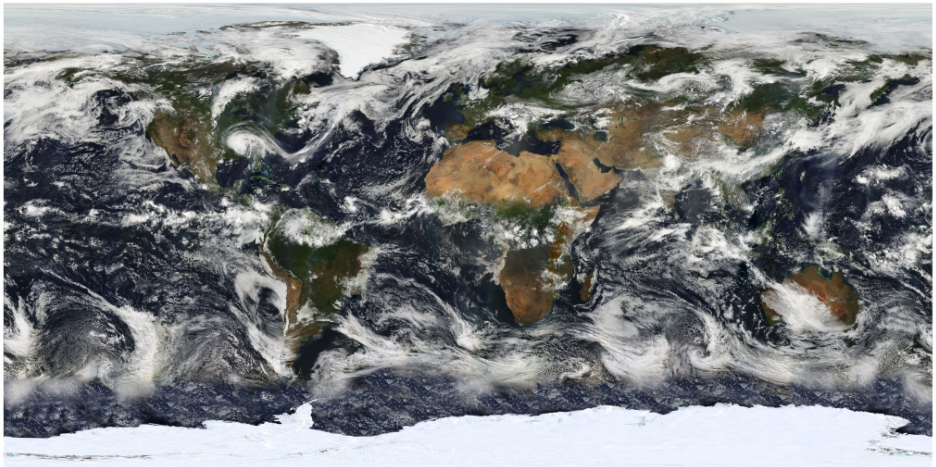
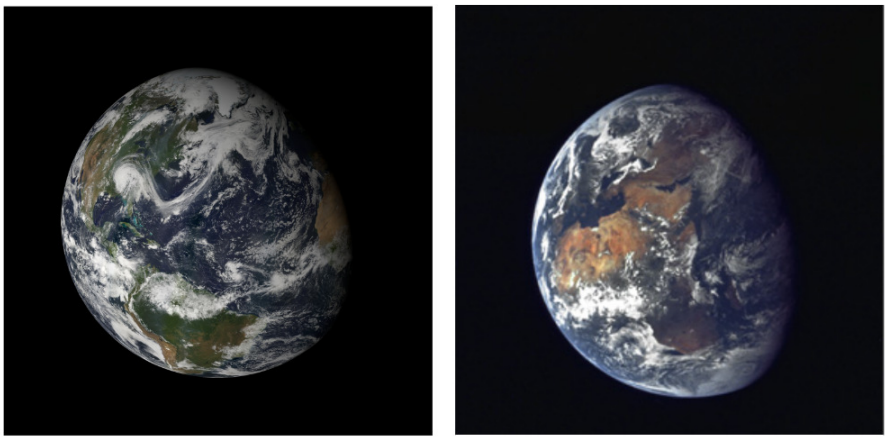


Figure 2.4: Initially chosen texture of the Earth

In figure 2.5 we can see a comparison of the synthetically generated image and an actual true image of the Earth as seen from the Apollo 11. From a first glance, used texture gives a good representation of what we should see when we look of a picture of the Earth taken from the space.



(a) POV-Ray representation of the Earth (b) Apollo 11's picture of Earth

Figure 2.5: Comparison of synthetic Earth image with Apollo 11's picture (the two images have different resolutions)

Despite the relatively good result, this way of modeling the Earth still haves some issues :

- since for all pictures we use the same texture, the clouds are sticked to their position on top of the surface of the Earth;
- terrains and seas are characterized by the same optical properties, while in reality they are not;
- diffusion of sunlight in the atmosphere is not simulated;

In the following paragraphs, those issues will be addressed.

Cloud Layer
The fixed coupling of Earth surface and clouds is big problem for what concerns CV algorithms development, which should be trained to work in several different conditions, so, having the clouds always on the same region of the Earth despite the orientation is not acceptable. The solution adopted in both this work and [29] relies on decoupling the Earth terrain layer and the cloud layer, specifying different optical properties for each layer, specify clouds relative rotation with respect to the terrain and coupling back everything together. Furthermore, oceans have been split too, in order to be able to set different optical properties for the seas. For the terrain layer, a a true-color mercator image of the Earth has been used as a base, in this way every piece of the surface could possibly be visible in any picture.

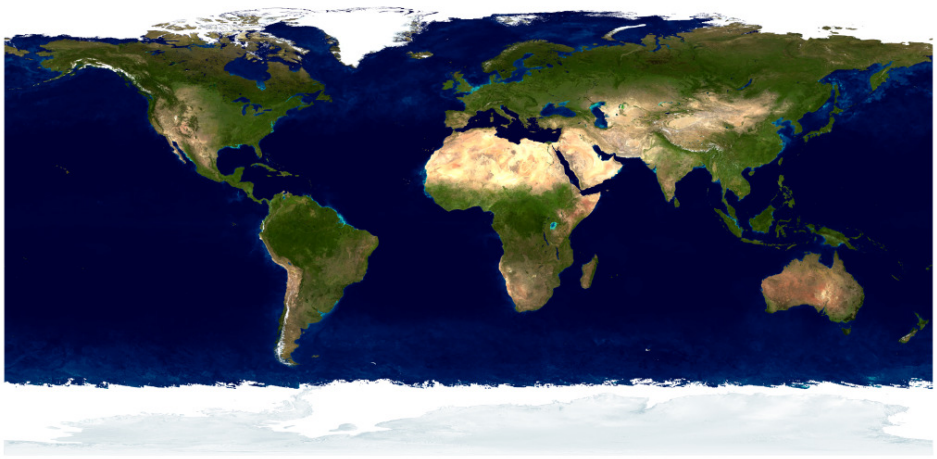


Figure 2.6: True Color Earth Mercator Image [46]

As said before, in order to be able to use different optical properties for water and terrains, a separate two-color mercator-projection image of the Earth where the water is black and the land is white has been used.

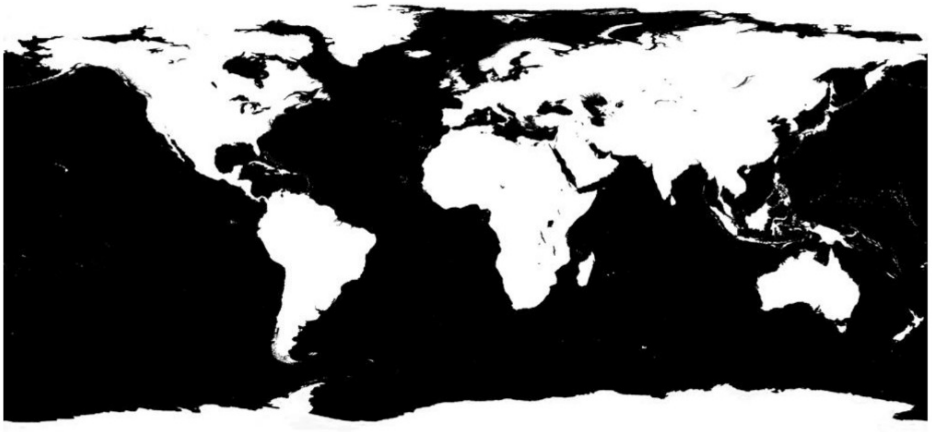
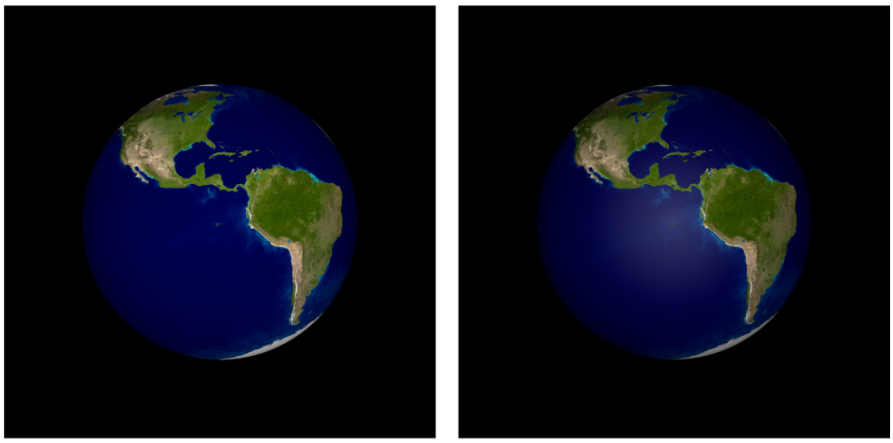


Figure 2.7: Landmask Mercator Image

In figure 2.8 the result of differentiating the optical properties for terrains and oceans is shown. Despite the fact that it may seems that there is only a small distinction between the two images (in particular, shinier oceans and darker forests), is still enough the make the Earth to seem more photo-realistic, and it leave some space for further tuning.

The cloud layer is added on top of the cloudless surface and thanks to that, it can rotate with respect to the Earth by a prescribed angle set by the programmer. The cloud layer texture and the shape of the clouds itself always remains the same, but this can be partially mitigated by using different cloud textures.

The cloud texture used for this work (figure 2.9) is not actually directly printed



(a) No differentiation of parameters (b) Differentiation of parameters

Figure 2.8: Comparing images with no differential treatment between land and ocean and with differential treatment

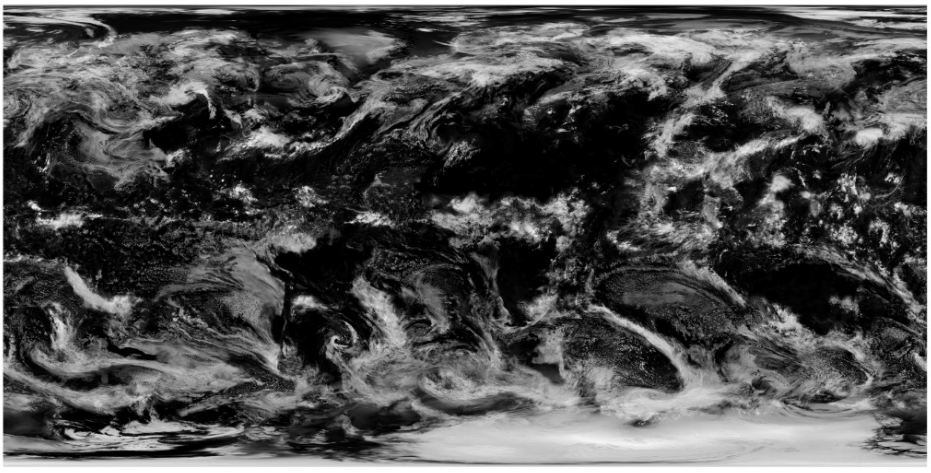


Figure 2.9: Two-color mercator image of Earth's cloud layer

on the surface of the earth, but rather is extruded on a shell built around the sphere which defines the Earth, which has an inner radius equal to $R_{in} = 1.001 \cdot R_{Earth}$ and an outer radius equal to $R_{out} = 1.0002 \cdot R_{Earth}$. Those values have been found by taking as a reference the fact that low Earth clouds ranges from an altitude of 600 m to 15 000 m [15]. Although higher or thicker clouds can be modeled, this will require longer rendering times. Of course, also for the cloud layer is possible to set custom optical properties, in order to make the clouds partially transparent, so that when there isn't a dense cloud area, the terrain behind is still visible under the white blanket.
The result of adding the cloud layer can be seen in figure 2.10

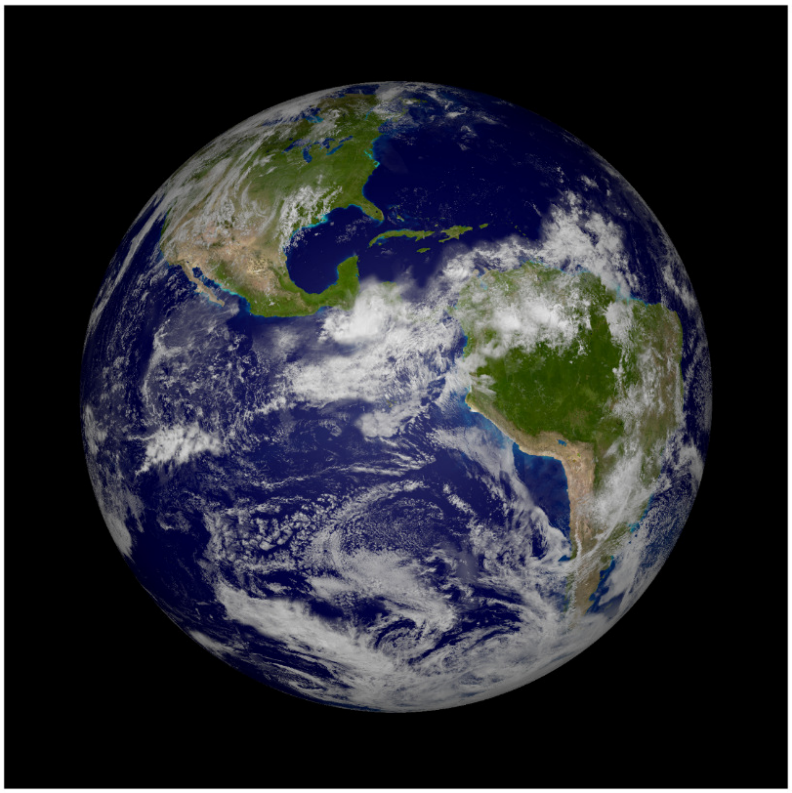


Figure 2.10: Earth's representation using cloud shell

Cloud Layer

Recreating the characteristic atmosphere presence around the Earth is of crucial importance for developing a CV algorithm for space applications, as it is needed in order to train the algorithm itself to work into a real-case scenario. The atmosphere's presence in fact would enlarge the aspect of the Earth in the image, and furthermore would stress the edge detection algorithms. To recreate the atmospheric shine effect, a strategy which is similar to the one adopted to model the cloud layer has been used. In fact, it has been modeled as a shell with a certain

thickness of a transparent material with some scattering properties. For what concerns both this project and what has been done in [29], the gaseous layer was made only 25 km thick. Despite the fact that the atmosphere should be visible up to many more kilometers, the computational load introduced by rendering a much high atmosphere is not negligible on a standard PC hardware, and so the image generation time would grow up exponentially, making the task of compiling a data-set of thousands of images very time consuming. In figure 2.11 can be seen the final result of adding the atmospheric model.

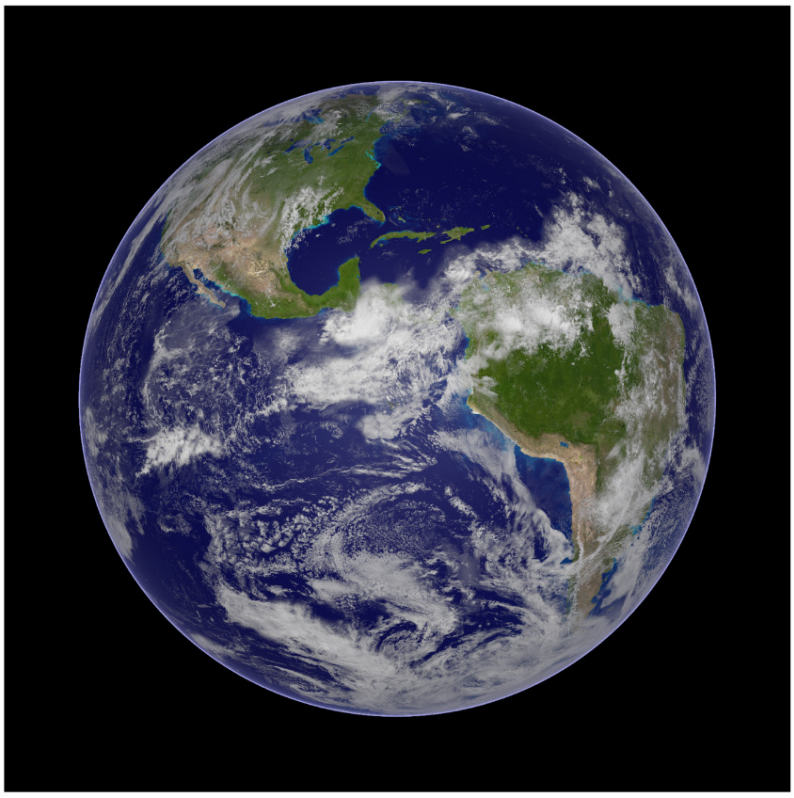
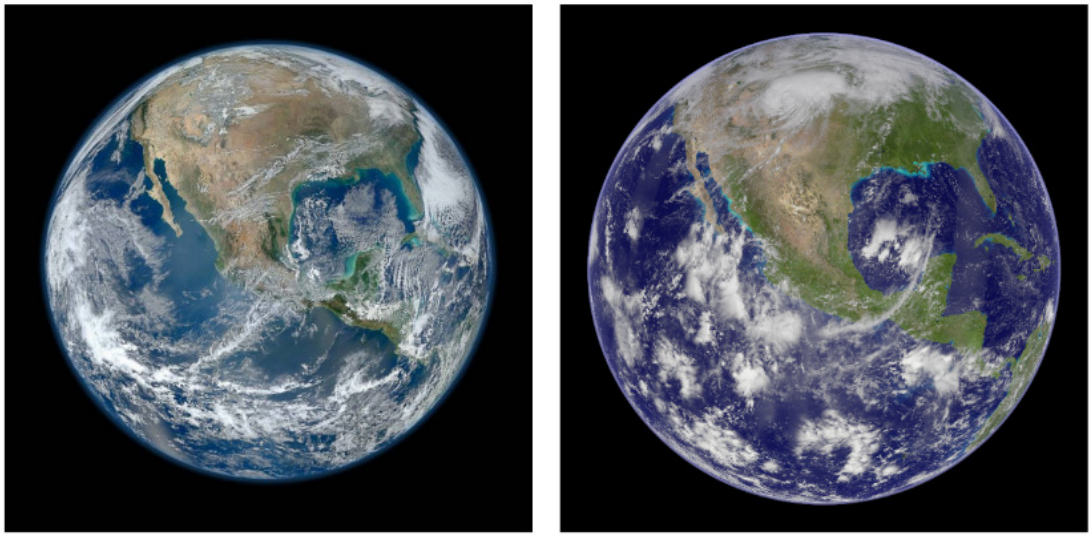


Figure 2.11: Earth with the atmosphere layer

In figure 2.12 instead is possible to view the artificially generated next to a real Earth picture taken by NASA's Suomi NPP on January 4, 2012. It can be seen that, despite the fact that the real image isn't perfectly reproduced (because some other factors should be known in order to be taken into account, such as exposure time), the synthetic image still provide an high degree of similarity with the real one.

In table 2.1 are briefly resumed the values used to model the various layer of the Earth.



(a) True Earth image (b) Synthetic Earth image

Figure 2.12: Comparison between true [46] and final rendered Earth image

	Terrain	Oceans	Clouds	Atmosphere
Ambient	0.001	0.001	0.001	0.0001
Roughness	0.05	0.1	0.005	0.5
Brillance	1	1	1	1
Diffuse	0.85	0.85	1	0.6
Reflection	0	0.04, 0.25	0	0
Specular	0	0.1	0	0

Table 2.1: Optical parameters of Earth's different layers

Light Modeling

For an object to show up into the scene, it must be illuminated. There are two ways to illuminate an object with POV-Ray:

- use a standard light source
- use ambient light

The light source is controlled by the keyword **light_source**, which in turn accepts several modifiers. Light sources in POV-Rayhave no visible shape of their own. They are just points or areas which emit light.
The ambient light instead is controlled by the keyword **ambient** added to the **finish** modifier of an object, and it is used to simulate the light inside a shadowed area. We can think of ambient light like a kind of light that is scattered everywhere in the room. It bounces all over the place and manages to light objects up a bit even where no light is directly shining. In our particular case, we can use the **ambient** option to simulate the illumination of the object due to spurious light sources (such as stars) or reflection from other bodies (like the Moon or other planets). In order to model the lightning condition of a true solar system, the light source has been modeled to resemble as much as possible the light emitted by the Sun. The solution adopted in this project and in [29] relies upon modeling the sun as an area light source through the option **area_light**. This allows to create a cluster of point-like light sources distributed on a disc (simulated by adding the circular option to the area light source) which has radius equal to the radius of the Sun, and placed at the exact distance which the Sun has from the Earth in the ECIF frame. In order to cope with the fact that in reality the Sun is equal to a sphere of light and not a disc, the option orient has been used. When using orient, every object in the POV-Ray world would see the Sun's disc as oriented toward it, from any position around it. Furthermore, the option **jitter** has been used, which tells the ray-tracer to slightly move the position of each light in the area light to eliminate any shadow banding that may occur.

2.2.3 Spacecraft Modeling

3D Model of the Spacecraft

The problem of modeling rather complex shapes with POV-Ray, such as can be a spacecraft, it was one of the most demanding parts of this work, which also required to patch POV-Ray source code, to prevent the ray tracer to forcibly use unnecessary features. Obviously, building the S/C using POV-Ray primitives is a sub-optimal solution, as many different S/C can have different shapes, and modeling them by using simpler shapes each time it is simply not efficient. Furthermore, usually detaled 3-D CAD models of S/C are available which can be easily exported into STL format, so the focus has been putted into trying to understand how

to translate those 3-D CAD models directly into POV-Ray code. Several open source and closed source software have been coupled together in order to produce trough POV-Ray a render of a given STL model. The procedure will be detailed in the following paragraphs. The S/C which has been modeled is heavily inspired by the Tango S/C of the PRISMA mission [2], by ESA. The spacecraft has been modeled using the dimensions specified in [60], which will be here reported for ease of reading. The solar panel is represented by a polygon of 570 mm x 759 mm, while the spacecraft body instead is represented by a convex polyhedron of 560 mm x 550 mm x 300 mm. The radio frequency antennas length instead is of 204 mm. The origin of the CAD model is located in correspondence of the CG.
Using the aforementioned dimensions, the Tango spacecraft has firstly been reproduced using Autodesk Inventor. The result of the modeling procedure is shown in figure 2.13.

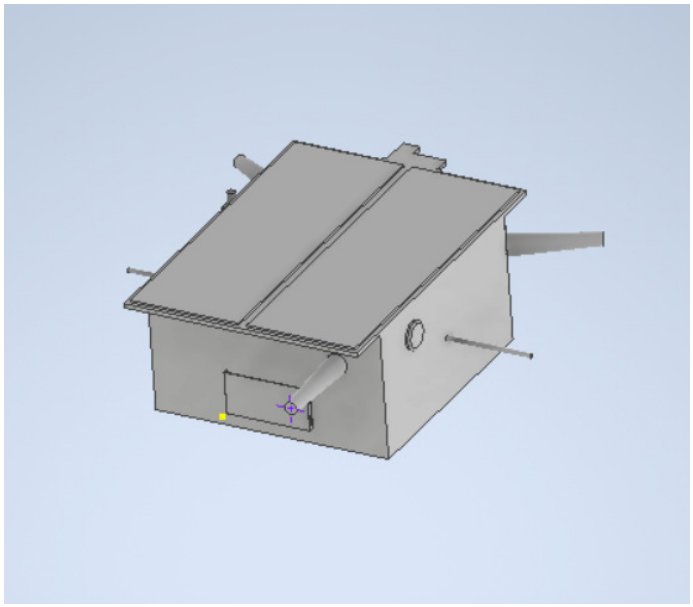


Figure 2.13: Tango S/C 3-D model

The 3-D CAD model can now be exported in STL format to be manipulated through other software.

Blender

Once the 3-D model of the Tango S/C was available, the most challenging task has been the one of rendering the S/C itself using POV-Ray at attitudes imposed by the user. The first step for archiving that goal is to import the STL model of the S/C into Blender. By exploiting the POV-Ray render add-on for Blender, it is

[2]More details about the PRISMA mission can be found at `https://earth.esa.int/web/eoportal/satellite-missions/p/prisma-prototype`

possible to render any given STL file imported file using POV-Ray as rendering engine. The POV-Ray add-on will optionally save the generated SDL code used to render the scene. The generated code however, will threat the entire 3-D model as a whole, generating one giant POV-Ray mesh2 object, that is something which cannot be easily managed. Just as an example, would be impossible to assign to the different parts of the S/C different optical parameters or textures. So, to workaround this limitation, it is a good practice to first split the different surfaces of the 3-D model in different different children objects (or children surfaces), and only after render the scene in order to get the POV code. In this way, the POV-Ray render add-on for Blender will generate a **mesh2** object for each children object created in Blender. This will enable us to set different material properties for each children surface or to apply different textures to different surfaces. For the purpose of this work, the original STL model has been subdivided into thirty children object, each one with its own optical parameters, as can be seen from figure 2.14 .

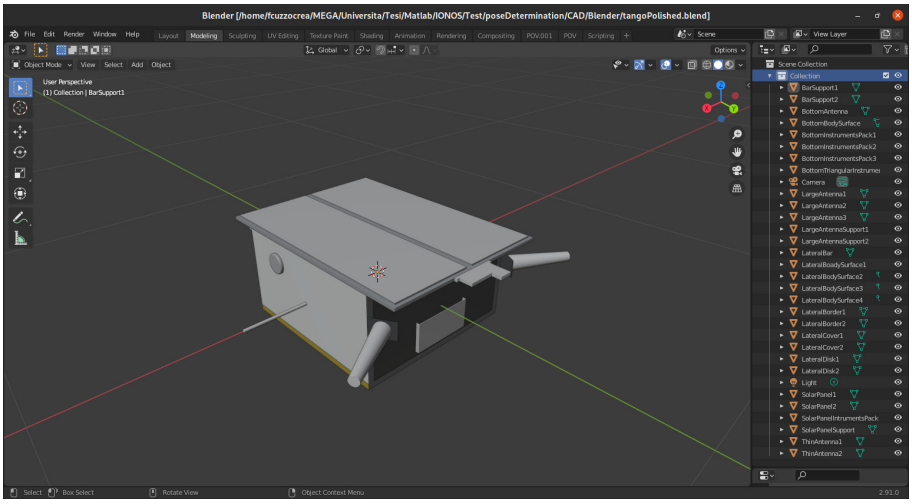


Figure 2.14: Tango S/C 3-D model in Blender

The Blender POV-Ray add-on also let the user to inject custom POV code (figure 2.15) into the auto-generated one, which can be useful especially for adding textures, for example to the solar panels.

The end results is showed in figure 2.18

POV-Ray

The major issue of the code generated from the POV-Ray Blender add-on is that is composed of several separated **mesh2** objects which makes almost impossible to rotate the whole object by imposing a given attitude matrix, since one is supposed to rotate all the **mesh2** objects by hand.

In order to workaround this limitation, from all the **mesh2** objects which have been generated from POV-Ray Blender add-on are merged into one single **merge**

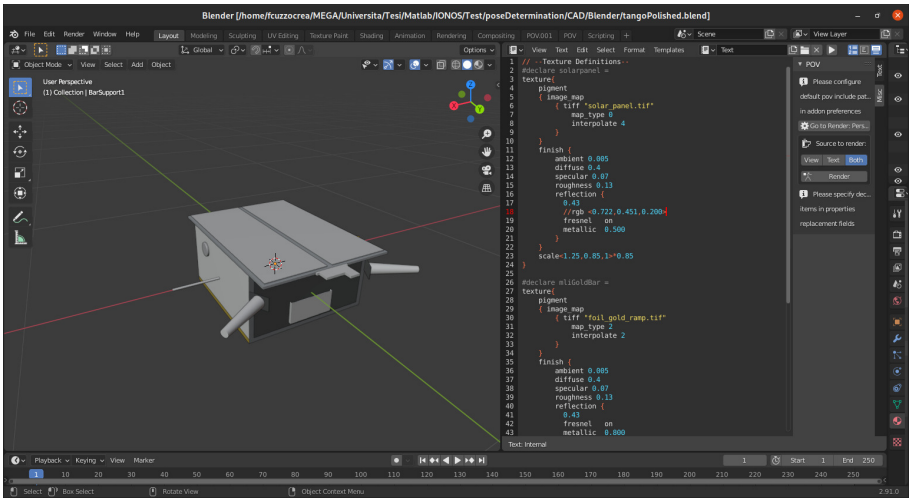


Figure 2.15: Add custom POV code into Blender

Figure 2.16: Texture used to model MLI

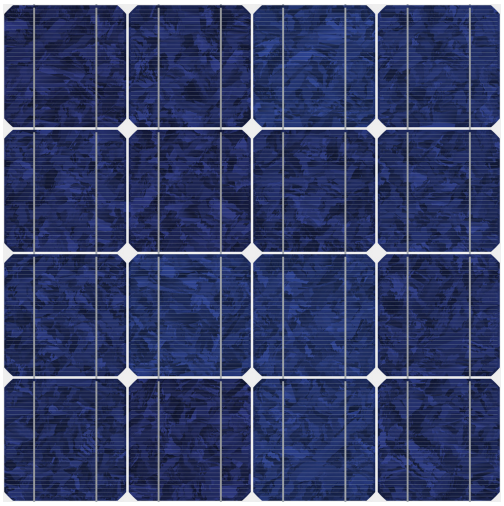


Figure 2.17: Texture used to model solar panels

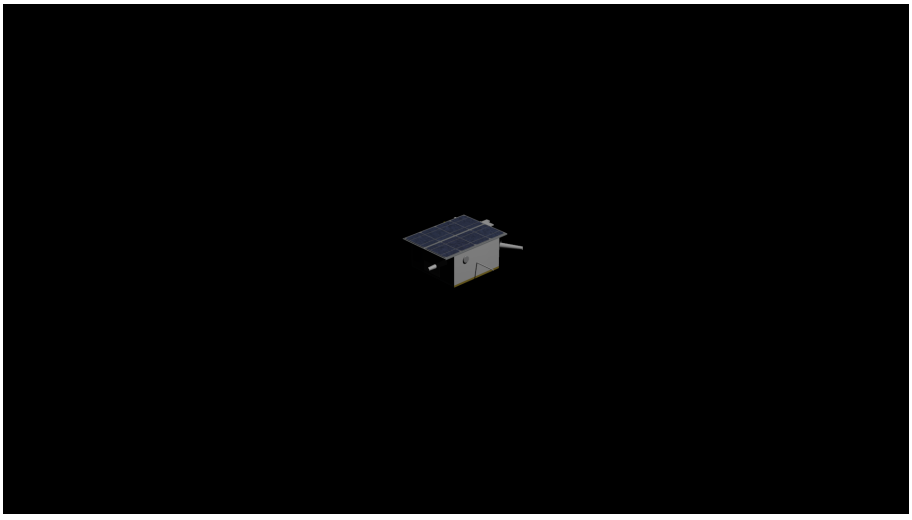


Figure 2.18: Tango S/C rendered using POV-Ray Blender add-on

object. The **merge** POV-Ray operation allows to bind two or more shapes into a single entity that can be manipulated as a single object, which is exactly what we want. The new object created by the merge operation can be scaled, translated and rotated as a single shape. The entire merge can share a single texture and optical parameters but each object contained in the union may also have its own texture and optical parameters, which will override any texture statements in the parent object. So, all the **mesh2** objects which are describing the S/C surfaces are merged into a single big (21K LoC) **spacecraft merge** object. To ease the usage of the **merge** object, a POV-Ray include file it is created, with the sole purpose of containing the **spacecraft merge** object. The include file is read in as if it were inserted at that point in the file. Using include is almost the same as cutting and pasting the entire contents of this file into the scene. This allow to define the **spacecraft merge** once and call it from any other POV file just like any other predefined object is called, and so, it is possible to manipulate its position and orientation in a much more easier way by just using the **matrix** keyword. The **matrix** keyword allows to specify directly the orientation of the **spacecraft** object, $\mathbf{A_{TN}}$, and its location with respect to POV-Ray "inertial" world. In table 2.2 are briefly resumed the optical parameters used to model the different part of the S/C.

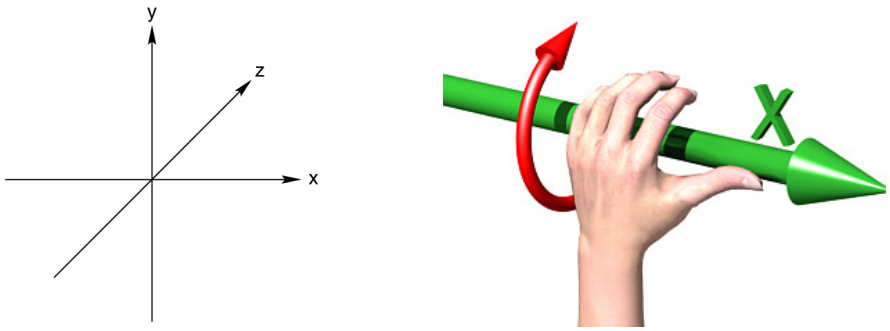
	Solar Panels	Antennas	Main Body
Ambient	0.0	0.25	0.25
Roughness	0.13	0.0005	0.0005
Brillance	-	3.15	3.15
Diffuse	0.3	0.95	0.99
Reflection	0.23, 0.5	0.65, 1	0.65, 1
Specular	0.04	0.96	0.96
Phong	-	0.43	0.43
Phong Size	-	25	25

Table 2.2: Optical parameters of S/C different parts

2.2.4 Camera Modeling

POV-Ray allows to simulate a perspective pinhole camera (more details about the pinhole camera will be given in section B.1). In POV-Ray's camera environment, the programmer can set all relevant camera parameters, which will be then used to simulate the camera trough which the scene will be rendered. The most meaningful parameters which can be imposed are the location of the camera itself, the direction of the boresight axis (where is the camera looking at), the view angle and the direction of the camera reference frame.
The major issue which has been faced when modeling the camera in POV-Ray is

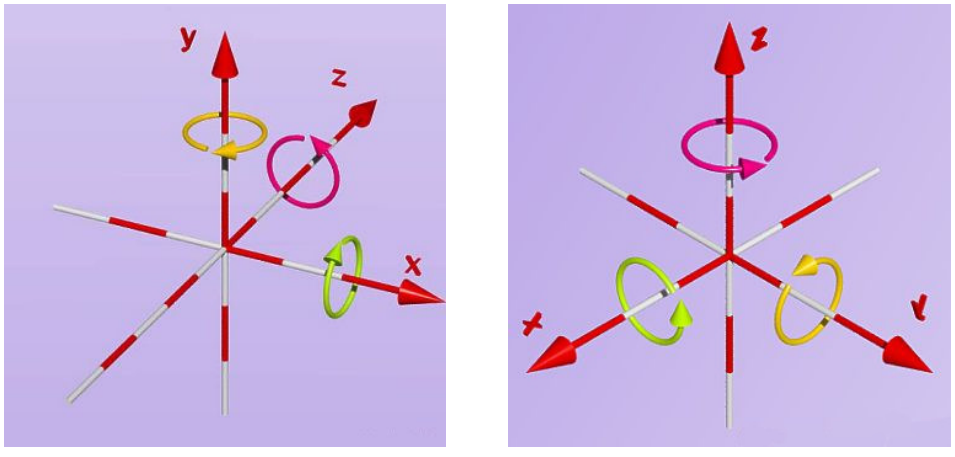
the fact that the program defaults to a left handed coordinate system to describe the scene, while all other software (like Autodesk Inventor, Blender) are using a right handed coordinate system.



(a) POV-Ray axis directions (b) Left hand rule

Figure 2.19: POV-Ray coordinate system

Moreover, the right handed coordinate system is used also to model the orbit that the spacecraft will follow and the spacecraft attitude from Euler equations. It is possible to trick POV-Ray to behave like it using a right handed coordinate system by acting on the **right** vector of the camera environment. The camera **right** vector describes the direction to the right of the camera, so, in practicte, tells POV-Ray where the right side of the screen is. Therefore, the sign of the x component of the **right** vector can be used to determine the handedness of the coordinate system in use.

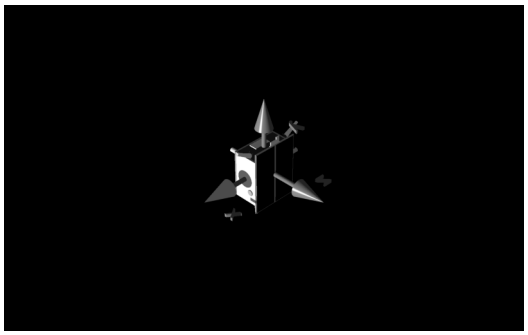


(a) Left Handed Coordinate System (b) Right Handed Coordinate System

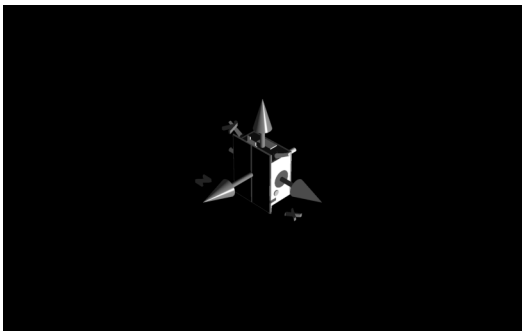
Figure 2.20: Left Handed Coordinate System and Right Handed Coordinate System

By default POV-Ray use a positive x value in the **right** vector. This means that the right side of the screen is aligned with the +x-direction. By using a

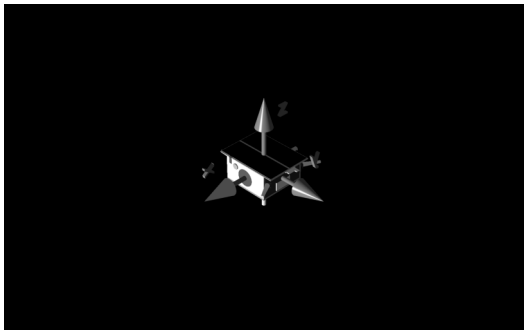
negative x value in the **right** vector instead the right side of the screen will be aligned to the -x-direction, and the coordinate system will be right handed. Doing only that however left us having the y axis as the one pointing upward and the z axis as the one point in the direction which goes outside the screen. To have a more comfortable reference frame, aligned with the ECIF reference frame introduced in section 2.1, we can flip y and z axis by overriding the **sky** vector. By default, in fact, POV-Ray uses <0,1,0> as **sky** vector. By redefining it as <0,0,1> POV-Ray will roll the camera until the top of the camera is in line with the **sky** vector, giving us the desired reference frame.



(a) POV-Ray standard coordinate system



(b) Result after imposing negative x to the **right** vector



(c) Result after imposing <0,0,1> to the **sky** vector

Figure 2.21: Reference Frame Rotations

In order to rightly simulate a real camera, the POV-Ray camera aperture angle has been computed by assuming the intrinsic properties of a real camera, a Point Grey Grasshopper 3, equipped with a Xenoplan 1.9/17 mm lens, which is the same camera used to capture the real images of the SPEED data-set [36].

By assuming a square CCD sensor with square pixels we can obtain:

$$A.R. = \frac{N_u}{N_v}\,, \tag{2.1}$$

Parameter	Description	Value
N_u	Number of horizontal pixels	1920
N_v	Number of vertical pixels	1200
f_x	Horizontal focal length	17.6 mm
f_y	Vertical focal length	17.6 mm
d_u	Horizontal pixel length	5.86×10^{-3} mm
d_v	Vertical pixel length	5.86×10^{-3} mm

Table 2.3: Parameters of the camera used to capture the SPEED images [36]

$$CCD_{size} = d_u \cdot N_u \,, \tag{2.2}$$

$$\alpha = 2 \cdot \arctan\left(\frac{CCD_{size}}{2 \cdot f_x}\right), \tag{2.3}$$

where $A.R.$ is the Aspect Ratio of the picture, which is passed as first component **right** vector (with the negative sign, as described before), and α is the aperture angle which will be input in POV-Ray. Using the parameters detailed in table 2.3 we can compute:

- $A.R. = 1.6$;
- $\alpha = 35.4°$.

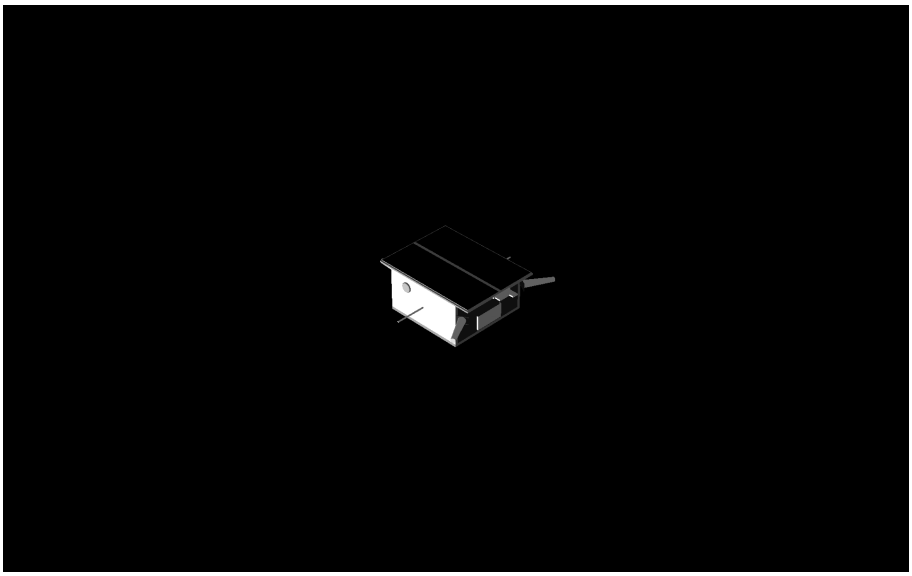


Figure 2.22: Tango S/C rendered using table 2.3 parameters

Camera Attitude

Being able to correctly know the imposed camera attitude at the moment of image generation is of crucial importance, especially when developing images to test pose determination algorithms. Using the **look_at** keyword we specify the direction of the camera's boresight axis (so, in practice, where the camera is looking at, as the keyword itself suggest). Therfore, by imposing the **look_at** vector, we are implicitly imposing the camera attitude with respect to what is looking at. For what concerns this work, it is assumed that the camera always looks at the center of the target S/C. By imposing **sky** and **right** vectors we impose the initial direction which POV-Ray uses for orienting the camera. By imposing the **look_at** vector POV-Ray will first roll the camera until the top of the camera is in line with the sky vector. Then it uses the **sky** vector as the axis of rotation left or right and then to tilt up or down in line with the **sky** until until it lines up with the **look_at** point. Then tilts straight up or down until the **look_at** point is met exactly.

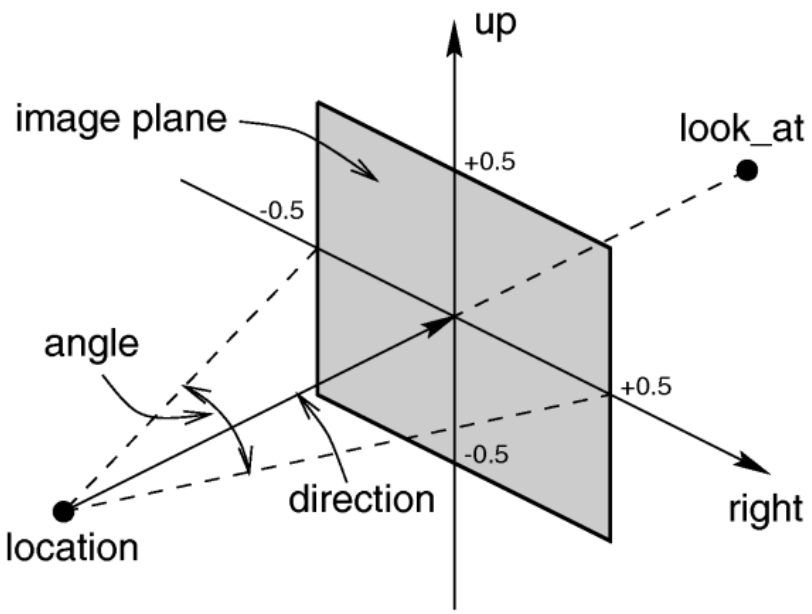


Figure 2.23: POV-Ray default perspective camera

So, by knowing how POV-Ray rotates the camera to point to the the **look_at** point, we can compute the actual attitude matrix of the camera with respect to the ECIF frame[3] exploiting simple linear algebra rules:

- the z-direction, which is the one going straight out from the camera, can be

[3]It is reminded to the reader that the ECIF reference frame and the POV-Ray's world frame are aligned by design

computed by simply making the difference between the **look_at** vector and the camera **location** vector;

- the x direction, the one going right on the image plane, can be found by the cross product between the z-direction and the **sky** vector;
- the y direction, which is the one going downward in the image plane, can be found by performing the cross product between the z-direction and y-direction.

So, doing the needful math it is possible to write:

$$\hat{\mathbf{z}} = \frac{\mathbf{look\_at} - \mathbf{location}}{\|\mathbf{look\_at} - \mathbf{location}\|}\,, \tag{2.4}$$

$$\hat{\mathbf{x}} = \frac{\hat{\mathbf{z}} \wedge \mathbf{sky}}{\|\hat{\mathbf{z}} \wedge \mathbf{sky}\|}\,, \tag{2.5}$$

$$\hat{\mathbf{y}} = \frac{\hat{\mathbf{z}} \wedge \hat{\mathbf{x}}}{\|\hat{\mathbf{z}} \wedge \hat{\mathbf{x}}\|}\,. \tag{2.6}$$

Knowing the versors which define the reference frame, the attitude of the camera with respect to the inertial POV-Ray frame, $\mathbf{A_{CN}}$, can be simply defined as :

$$\mathbf{A_{CN}} \triangleq [\hat{\mathbf{x}}, \hat{\mathbf{y}}\,.\hat{\mathbf{z}}] \tag{2.7}$$

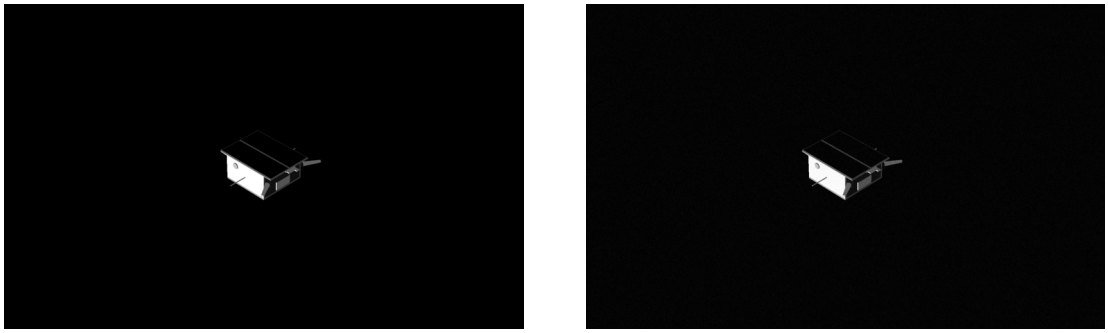
At this point, the imposed ground truth pose ($\mathbf{A_{TC}}$, $\mathbf{t_C}$) of the camera with respect to the S/C is completely defined by:

$$\mathbf{A_{TC}} = \mathbf{A_{CN}}\mathbf{A_{TN}^{T}}\,, \tag{2.8}$$

$$\mathbf{t_C} = -(\mathbf{look\_at} - \mathbf{location})\mathbf{A_{TN}^{T}}\,. \tag{2.9}$$

2.2.5 Noise Modeling

Finally, for augmenting the "realness" of an image generated though POV-Ray, the image needs to be post-processed using MATLAB. The image so is input in MATLAB and two different noises are applied through the imnoise command : speckle noise ($\sigma^2 = 0.004$) and Gaussian white noise ($\sigma^2 = 0.003$). The final result can be seen in figure 2.24.



(a) Image without noise added

(b) Image wit speckle and Gaussian white noise

Figure 2.24: Comparison between image without noise and image with speckle and Gaussian white noise

2.3 MATLAB Integration

Since it would be highly unpractical to hand edit every time all the parameters to render the S/C in different position and attitudes as well as Earth's different rotations, a MATLAB toolbox has been developed which allows the user to automatically generate .pov SDL files, and it renders them trough POV-Ray automatically. The implemented toolbox takes as input N_u, N_v, f_x, f_y, d_u, d_v to compute intrinsic camera parameters to set POV-Ray camera. Then, it offers the user the possibility to generate a sequence of random attitudes [3], given the orbital parameters of a reference orbit. If inertial properties of the target S/C are available, instead, it can compute the full uncontrolled dynamics (so to speak, the dynamics of the S/C when only subjected to the disturbances torques acting on the reference orbit) of the S/C trough a Simulink model at each time instant. For the interested reader, details about how the Space environment has been modeled in Simulink are given in A. For each time instant for which the attitudes of the target S/C have been computed, the relative position of the camera with respect to the target S/C is computed by selecting a random point inside a sphere having a 20 m radius centered in the target position, and the imposed pose is computed. A random Earth's rotation is computed as well. All the properties of the scene are stored in a custom .att file, which can be passed in a second time to retrieve the absolute position and orientation of both the camera and the target S/C and their pose. A .pov SDL file is written for each time instant for which the attitudes of the target S/C have been computed and POV-Ray is called to render each image in the background, from within MATLAB. Rightly after an image has been rendered, it is post-processed by MATLAB in order to add the noise, as described in 2.2.5.

2.4 Conclusions

It must be noted by the reader, that a patched POV-Ray version is required in order to correctly replicate what has been done in this project, as stated in the first lines of section 2.2.3. At the beginning of this project it has been decided that by design, 1 unit in the POV-Ray world is exactly equal to 1 km to ease the modeling of the solar system. Thus, the Earth is placed at the origin of the POV-Ray world (consistently with the ECIF frame)and it has a 6378 km radius, and the light source, the Sun, is placed 1.4723×10^8 km far from the Earth, and has a radius of 696 340 km. The S/C instead is placed on an orbit characterized by the orbital parameters specified in table 2.4. The inertial properties of the target instead are computed by assuming a mass of 50 kg and using the dimension given in 2.2.3.

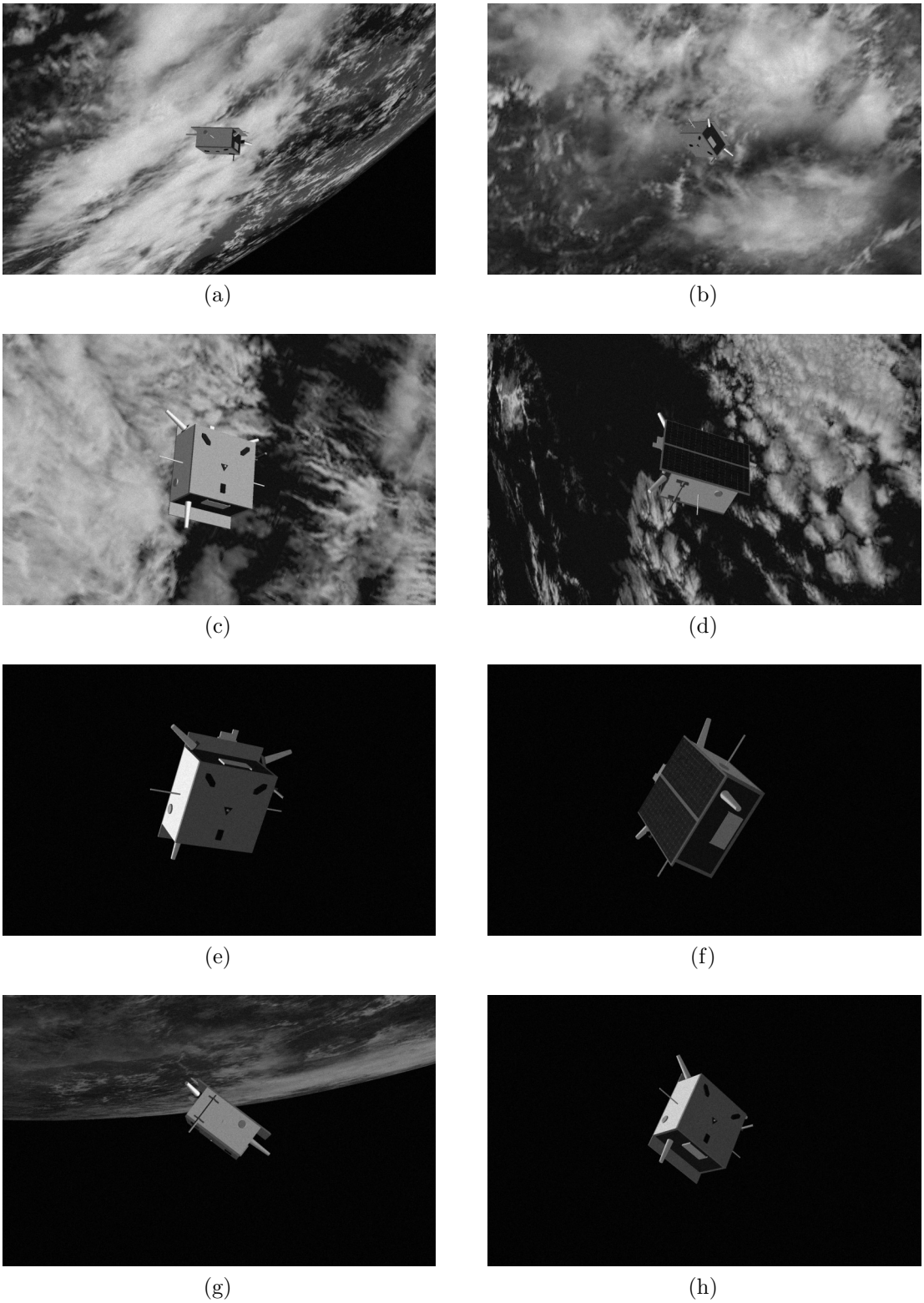
Orbital Parameters	Symbol	Values
Apogee	r_a	7178 km
Perigee	r_p	7101 km
Semimajor axis	a	7133 km
Inclination	i	98.28°
Pericenter anomaly	ω	0°
True anomaly	θ	0°
Eccentricity	e	0.0045°

Table 2.4: Parameters used to generate the reference orbit [48]

So, what is being represented is a scene having an extremely small object (the S/C) in front of a giant object the Earth, which is impacted by light which is generated from a very far distance (the Sun). Due to the exquisite peculiarity of this kind of scene, in some corner cases a "bounding issue" can be triggered, causing the S/C to be only partially redendered in the scene, or, in worst case scenario, not being rendered at all. Directly citing the POV-Ray documentation, in order to speed up the ray-object intersection tests POV-Ray uses a variety of spatial sub-division systems. The primary system uses a hierarchy of nested bounding boxes. This system compartmentalizes all finite objects in a scene into invisible rectangular boxes that are arranged in a tree-like hierarchy. Before testing the objects within the bounding boxes the tree is descended and only those objects are tested whose bounds are hit by a ray. This can greatly improve rendering speed. However, during the development of this project turned out that POV-Ray automatic bounding is not perfect. There are situations where a perfect automatic bounding is very hard to calculate, like the one faced in this project. Unfortunately, POV-Ray developers removed the possibility of turning off bounding control in POV-Ray, thus it is necessary to compile a patched POV-Ray version which introduces back the command line switch **-MB** which allows the user to turn off

bounding control. According to the GPL license POV-Ray is shipped with, the patch has been made freely available [4]. The output of the toolbox which has been described trough this chapter can be observed in figure 2.25.

[4] https://github.com/fcuzzocrea/povray/commit/2aecdfb20eef3ed3b6a5698392a9c91fa3f1afcd



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 2.25: Final result

Chapter 3

The SVD Architecture for Pose Initialization

"Prediction is very difficult, especially if it's about the future."

Niels Bohr

Performing a robust pose initialization from a 2-D image captured by a monocular camera in space is a very though task, particularly due to the fact that illumination conditions in space may vary during a single orbit, and therefore may cause inaccurate and unreliable features detection. Moreover, the Earth presence in the background introduces a major complexity into distinguishing the target S/C shape from the Earth surface. The SVD pose initialization architecture aims at solving the problem of pose initialization by employing a single 2-D image taken by a monocular camera. In this chapter, a detailed description of the SVD architecture implementation which has been carried out in this work will be presented.

3.1 Feature-Based Pose Estimation

The images generated trough the toolbox presented in Chapter 2 will now be analyzed using the robust monocular vision-based pose initialization algorithm proposed by S. Sharma, J. Ventura and S. D'Amico in [60]. The problem of pose initialization consists in computing the rotation matrix,$\mathbf{A_{TC}}$ and the translation vector, $\mathbf{t_C}$ that describes the transformation between the camera frame, C, and the target body frame, B. Given a generic image point, $p = [u, v]^T$, we can relate the corresponding $\mathbf{q_B}$ point of the 3-D model by employing the standard pinhole camera model (the interested reader can found a brief introduction to the pinhole camera model in appendix B.1):

$$\mathbf{r_C} = [x_C \quad y_C \quad z_C]^T = \mathbf{A_{TC}}\, \mathbf{q_B} + \mathbf{t_C}\,, \tag{3.1}$$

$$\mathbf{p} = \left[\frac{x_C}{z_C} f_x + C_x, \frac{y_C}{z_C} f_y + C_y \right] , \tag{3.2}$$

where f_x and f_y are the respectively the horizontal and the vertical focal length and $[C_x,\ C_y]$ are the coordinates of the center of the image. Moreover, it is assumed - without loss of generality - that the direction C_3 of the camera frame is pointed along the boresight of the camera and the directions C_1 and C_2 are aligned with the image frame defined by $P = (P_1,\ P_2)$

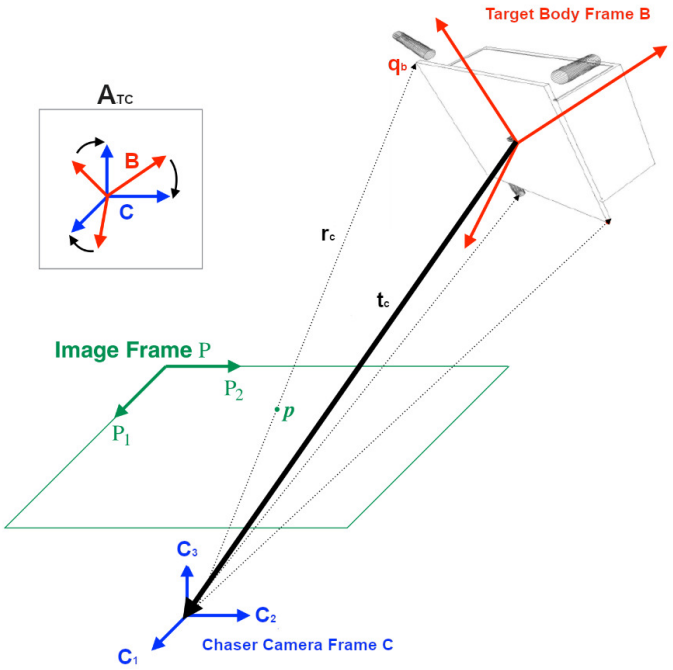


Figure 3.1: Schematic representation of the pose estimation problem using a monocular image [60]

As discussed in [16], we can build several considerations on top of those two coupled equations :

- the relationship between image points and pose parameter is hightly non linear and the problem of retrieving the 3-D points from the 2-D image can have infinite solution in under-constrained situations [24];
- the correspondence between $\mathbf{q_B}$ and $\mathbf{p}$ is not known *a priori*;
- the image coordinates need to be corrected for pixel's non-quadratism and lens distortion before using the perspective projection equations.

The pose estimation system thus needs the be capable of extracting the client features from the available image (image processing) to obtain measurements,

namely $\mathbf{p}$. The extracted features then needs to be matched to the correspondent elements of a client model (model matching), namely $\mathbf{q_B}$. The unknown pose then has to be estimated based on the available measurements (pose estimation). To be able to uniquely solve equations (3.1) and (3.2), at least six correspondences between the image points and the model points are needed. [24]. The challenging part of the problem is to be able to correctly detect the most meaningful points in the 2-D image. Usually this can be archived by employing edge detection techniques, such as the Canny edge detector [9] followed by the Hough transform [23]. This approach however may be biased it applied directly to the image due to the fact that these algorithm are gradient based and are not able to correctly distinguish the foreground from the background and also requires the definition of numerous hyper-parameters which are difficult to tune for broad applicability because the imaged scene and the illumination conditions are constantly changing throughout the orbit [60].

3.1.1 General Architecture

The architecture proposed in [60] tries to solve the pose estimation problem by:

- coupling the Weak Gradient Eliminator (WGE) technique with the Sobel edge detection to perform the image processing;
- using feature synthesis to reduce the search space for model matching;
- combining a P-n-P solver with the Netwon-Raphson (NR) method for pose estimation.

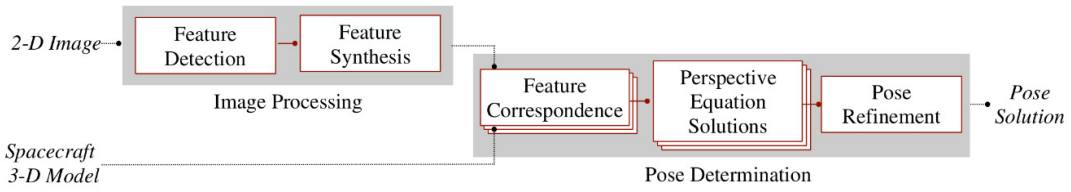


Figure 3.2: The SVD architecture for solving the pose estimation problem [60]

As the reader can see from the schematic represented in figure 3.2, the pipeline is composed from two different subsystems:

- the image processing subsystem, which receive as an input a 2-D image, distinguishing the S/C from the image, extract its edge features and groups them into geometric groups;

- the pose determination subsystems, which acceps as input the previously feature groups detected by the image processing subsystem and the 3-D model. It then pairs the 2-D and 3-D geometric groups to formulate multiple correspondence hypotesis. For each hypothesis formulated, the endpoints of the line segments forming the geometric groups are used to solve equation (3.1) and equation (3.2). The five best solution then are iteratively refined by employing the NR method.

Two are the most meaningful innovation the SVD architecture introduces:

- the usage of the WGE technique in order to distinguish the target S/C from the background and for enhancing the output of the edge detection procedure bu providing a robust identification of the small as well as large edges of the S/C;

- the usage of feature groups detected in the image and in the 3-D model to solve the feature correspondence problem, which also allows the SVD architecture to be more computationally efficient.

3.1.2 Image Processing Subsystem

The taks of the image processing subsystem is to extract the most meaningful features from the 2-D image a of the target S/C. To effectively and rapidly detect the target S/C edges - even in presence of the Earth in the background - an hybrid image processing subsystem is proposed in [60]. By fusing together the WGE technique and classical state-of-the-art edge detection techniques the architecture proposed by Sharma *et al.* is capable to provide a robust and efficient identification of the rue edges of the S/C. In particular, the WGE technique is able to identify the a Region of Interest (ROI) in a more accurate and robust way with respect to state-of-art techniques. The ROI is detection not only makes the entire subsystem robust against the background, but also allows an automated selection of the hyper-parameters needed for the Hough transform, which will be used to detect both small and large features of the S/C, in contrast with state-of-the-art algorithms which rely on a single set of manually tuned hyper-parameters. The flow of the image processing subsystem is illustrated in figure 3.3.

ROI Detection

The first thing to perform when analyzing an image is to identify a ROI. The raw input image - assumed to be rectified - is firstly imported in MATLAB. After that, A Gaussian filter is applied in order to the decrease the magnitude of image noise. The Gaussian filtering can be carried out by using the MATLAB function `imgaussfilt`. In this work, the images generated with the toolbox presented in chapter 2 are filtered by using a standard deviation $\sigma_X = 1.15$ before being fed to

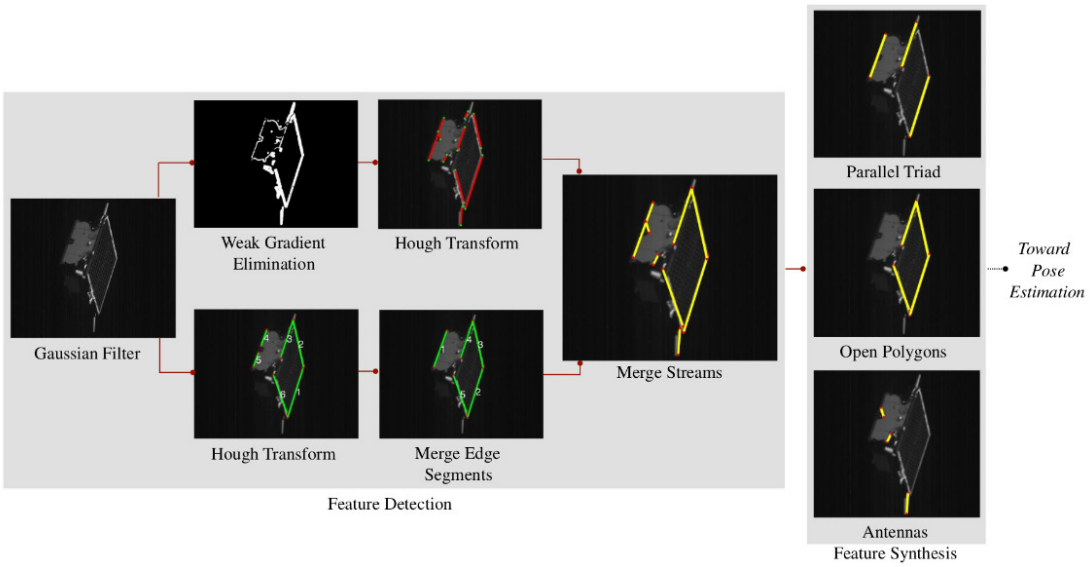


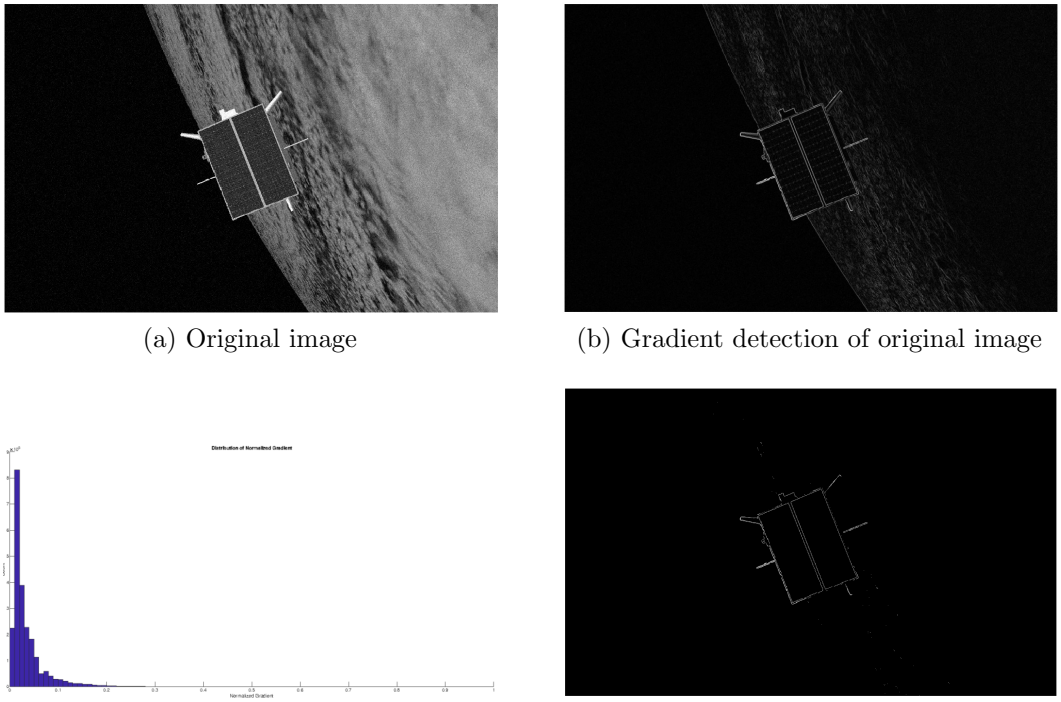
Figure 3.3: The image processing subsystem [60]

the image processing subsystem. During this work, has been observed that the choice of the σ_X value is very important for the success of the subsequent steps, as a wrong value can highly impact the effectiveness of the WGE technique. Once the image has been prefiltered, is fed as input to the WGE block, for the computation of the target's ROI. The first step consist in computing the magnitude of the image gradient, $|\nabla I|$, using the Prewitt filter for computing the image gradient (for more details about what the image gradient is, refer to appendix B.2). The S/C can be detected in the image by observing that, under most illumination conditions, in correspondence of the target S/C edges the gradient variation is more pronounced with respect to the background, regardless of the fact that the background is empty or the Earth is behind. Indeed, to detect the spacecraft, the gradient distribution is normalized using the MATLAB mat2gray function, sorted into 100 uniformly distributed bins and fitted by and exponential Probability Distribution Function (PDF), described by the equation :

$$y = \frac{1}{\mu} e^{-\frac{x}{\mu}} .$$

By observing the result histogram, it is clearly possible to see that most of the gradient intensities are weak and corresponds to the feature in the background or on the spacecraft surface. The weak gradient pixel locations can be classified by tresholding the PDF fit to the gradient histogram. The original authors suggest to use a value of 0.99 to treshold the PDF, however, for what concerns this work, all the pixel which corresponds to a cumulative distribution inferior to 0.996 are classified as weak and set to zero instead. Once the weak gradients are set to zero, only the most prominent features of the S/C are present in the image, as shown in

figure 3.4.



(a) Original image

(b) Gradient detection of original image

(c) Histogram of normalized gradient values

(d) Output filtered image gradients

Figure 3.4: Different steps of the WGE

By computing the Cumulative Distribution Function (CDF) of the vertical and horizontal gradient of the filtered image obtained by the WGE, we can determine the coordinates of a ROI where the features that yield the strongest intensity variations are located. Then, assuming that the filtered image gradient is normally distributed, the ROI coordinates can be found by limiting each of the previously computed CDFs between 0.025 and 0.975 in order to enclose the central 95% of the normal distribution.

As noted in [25], for images having a composite background the ROI detection is limited by the presence of interference, so a smarter choice for selecting the bounding box limits is to treshold the CDFs between 0.005 and 0.95. Therefore, only the central 90% of the normal distribution is selected. Restricting the range however negatively affect the ROI detection procedure for images were the background is empty. This problem can be solved by taking into account an additional constant, equal to the 5% of the mean edge length of the ROI for enlarging the ROI boundaries.

One of the innovative features of the WGE technique is that, once the ROI is correctly determined, it is possible to compute a coarse estimation of the the relative position and line of sight even before the pose determination subsystem.

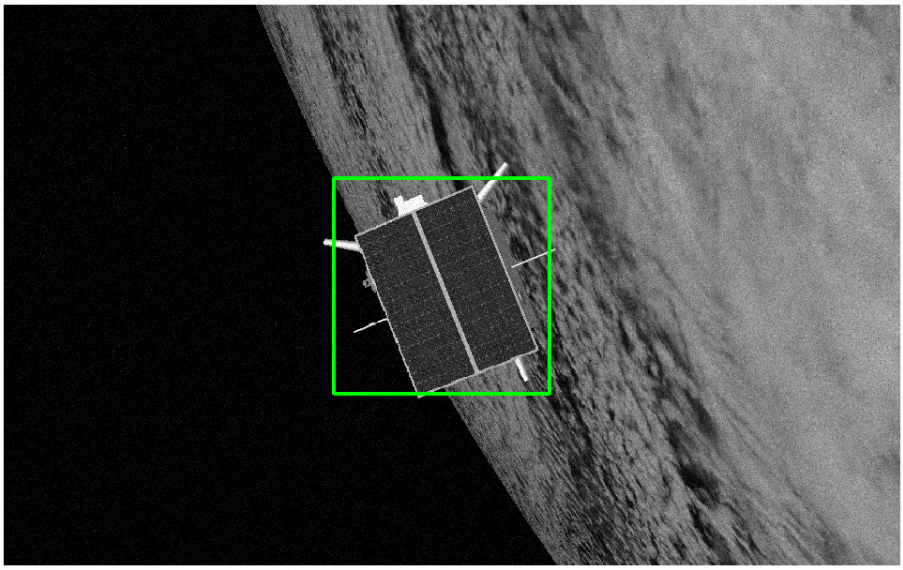
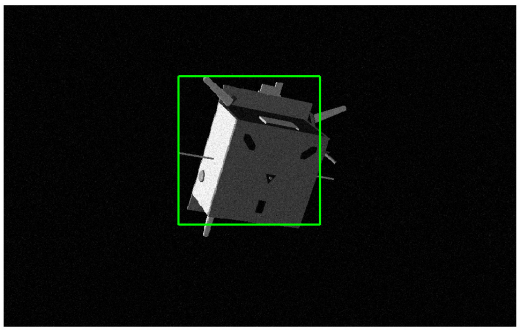
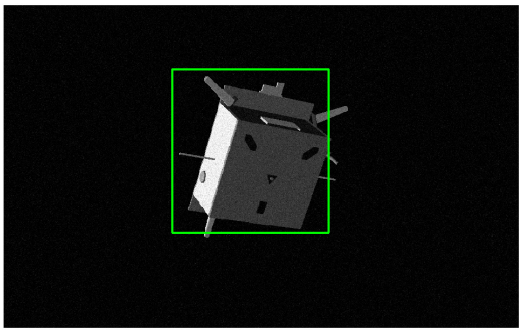


Figure 3.5: Result of the ROI detection procedure



(a) ROI selected by considering only the central 90% of the normal distribution



(b) ROI enlarged by adding the 5% of the mean edge length of the previously ROI

Figure 3.6: Improving the ROI detection procedure

By defining the diagonal length of the ROI, l_{ROI}, as:

$$l_{ROI} = \sqrt{{ROI_{width}}^2 + {ROI_{height}}^2}\,, \tag{3.3}$$

a coarse estimation of the range to the target S/C from the origin of the camera frame is given by :

$$\|t_C\|_2 = \frac{((f_x+f_y)/2)L_C}{l_{ROI}}\,, \tag{3.4}$$

were L_C denotes the diagonal characteristic length of the S/C 3-D model. By knowing the coordinates of the center of the image, $[C_x, C_y]$ and the coordinates of the ROI center, $[B_x, B_y]$, it is possible to compute the azimuth and elevation angles (α, β) from the origin of the camera frame, C, to the origin of the body-fixed coordinate system, B :

$$\alpha = \arctan\frac{B_x - C_x}{f_x}\,, \tag{3.5}$$

$$\beta = \arctan\frac{B_y - C_y}{f_y}\,. \tag{3.6}$$

Once (α, β) are known, the coarse relative position solution is given by :

$$t_C = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & \sin(\beta) \\ 0 & -\sin(\beta) & \cos(\beta) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \|t_C\|_2 \end{bmatrix}. \tag{3.7}$$

Feature Detection

The aim of the feature detection is to identify and extract from the input image a set of segments that corresponds to the true edges of the target S/C. In order to extract the features from the input image, a first edge detection process is applied to the tresholded image by means of the Hough transform. When applied to the problem of detecting straight lines into an image, the Hough transform performs a shape identification through a voting procedure in the parameter space (ρ, θ) (for more details about how the Hough transform work, the interested reader can refer to appendix B.3). When computing the Hough transform using MATLAB some parameters must be set:

- the desired resolution of ρ and θ,
- the maximum number of peaks to identify in the Hough transform matrix,
- the expected minimum length of the the line segment, $L_{min,Hough}$,

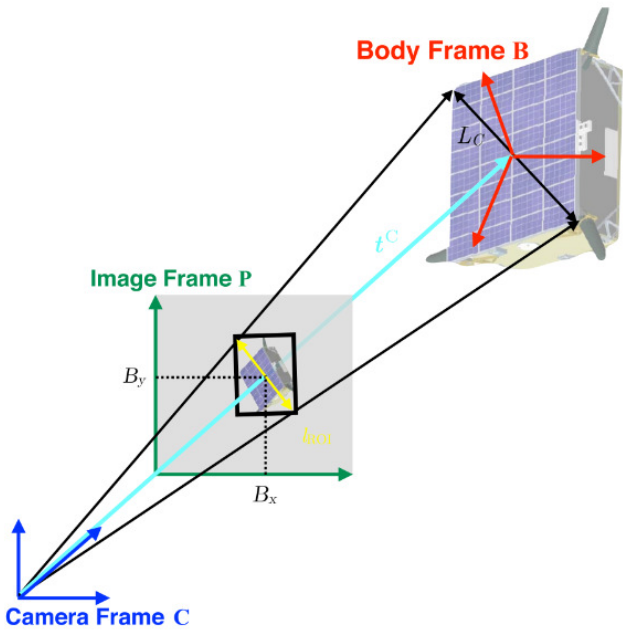


Figure 3.7: Calculation of a coarse relative position solution using the WGE technique [60]

- the expected minimum length of the the line segment, λ_{Hough} and the maximum gap between two points to be considered in the same line segment, λ_{Hough}.

The geometrical constraints imposed to the Hough transform are dictated by the choice of $L_{min,Hough}$ and λ_{Hough}, which are usually called hyper-parameters in the CV field. Fixing a value of ($L_{min,Hough}$, λ_{Hough}) for all the images will result into the inability of the algorithm to adapt to different distances between the camera and the target S/C. In fact, as the camera moves closer and closer to the target S/C, the latter will fill larger and larger portions of the image, and the expected lengths of its edges in the image plane are expected to grow, so a recomputation of the geometrical hyper-parameters would be needed. However, it is reasonable to hypothesize that the size of the detected ROI bounding box will grow too, so, in [60] is proposed to scale the geometrical hyper-parameters by using the information about the ROI diagonal length :

$$L_{min,Hough} = \kappa_1 l_{ROI} \qquad \lambda_{Hough} = \kappa_2 l_{ROI} \, ,$$

were κ_1 and κ_2 are two scalars which can be empirically estimated by performing several test simulations on ground.

The idea proposed in [60] is to tune the values of κ_1 and κ_2 in order to only extract short line segments from the tresholded image.

A second stream of features is then obtained by directly appling the Sobel operator, (for more details about what the Sobel operator is, refer to appendix B.2), to the unfiltered, rectified image, using the MATLAB function edge. The Sobel operator is applied by imposing an intensity treshold equal to 0.04. As proposed in [25], in order to eliminate the pixel chunks belonging to smaller reflective elements from the output of edge, it is possible to filter the output by using the MATLAB function bwareaopen which, once minimum pixel number P is defined, it removes all connected components (objects) that have fewer than P pixels from the input binary image. In contrast with what done in [25], which sets $P = 10$ for all images, in this work the P value is computed adaptively for each image exploiting the information about the ROI diagonal length :

$$P = \lfloor \eta \, l_{ROI}$$

where $\lfloor$ is the commonly used symbol for the **floor** operator and η is a multiplicative constants which can be tuned empirically. After that, the Hough transform is applied again on the output produced by the Sobel operator in order to extract line segments corresponding to the silhouette of the large components of the target S/C. The knowledge the ROI location, obtained through the WGE technique, is now exploited to automatically reject any line segment for which the midpoint lies outside the ROI itself. Also for computing the Sobel and Hough (S&H) stream of features, in [60] is advised to scale the geometrical hyper-parameters needed to compute the Hough lines by using the information about the ROI diagonal length :

$$L_{min,Hough} = \kappa_3 l_{ROI} \qquad \lambda_{Hough} = \kappa_4 l_{ROI} \, ,$$

where again, κ_3 and κ_4 are two multiplicative constants which can be fixed in an offline phase before the mission. The idea proposed in [60] is to tune the values of κ_3 and κ_4 in order to extract large line segments from the image.

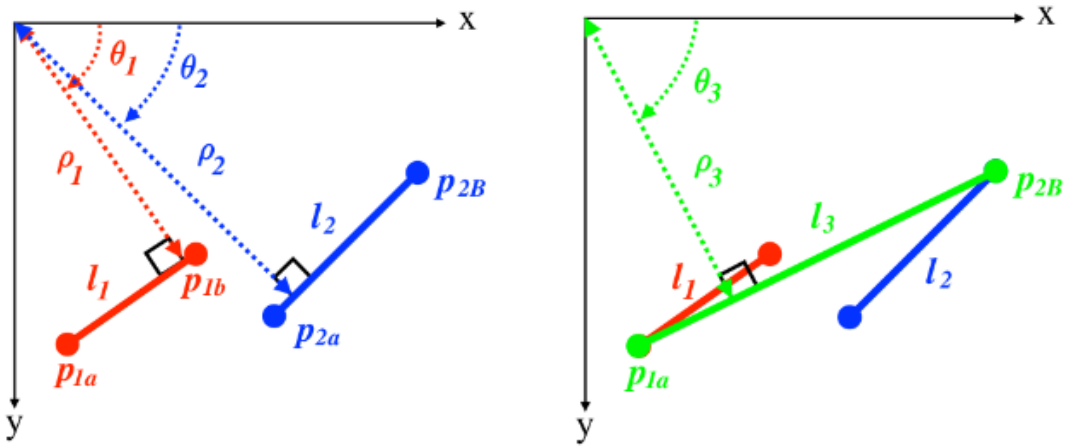
Merging Edges

The output of the Hough transform often results in multiple truncated edges, especially when applied to the image processed by the WGE. Since fragmentation can increase uncertainty and increase the computational cost of feature matching and pose solving [25], a section of the SVD architecture is dedicated to find and merge line segments which may corresponds to the same true line segment into a single line segment. To check whether two line segments can be considered similar and merged into one single edge, it is possible to define some geometrical checks on the basis of the parameters ρ and θ. Considering the two line segments represented

in 3.8a[1], it is possible to write their polar representation in the Hough space as:

$$l_1\rho_1 = x\cos(\theta_1) + y\sin(\theta_1)\,, \tag{3.8}$$

$$l_2\rho_2 = x\cos(\theta_2) + y\sin(\theta_2)\,. \tag{3.9}$$



(a) Original edge (b) Output merged edge

Figure 3.8: Merging two truncated edges [60]

The conditions of similarity proposed by [60] are then:

$$|\theta_1 - \theta_2| \leqslant \theta_{tresh}\,, \tag{3.10}$$

$$|\rho_1 - \rho_2| \leqslant \rho_{tresh}\,, \tag{3.11}$$

where θ_{tresh} and ρ_{tresh} are set to be equal to the resolution of θ and ρ in the Hough space. In [25] is proposed to scale the value of ρ_{tresh} using the information about the ROI diagonal length in order to improve the identification of the spacecraft true edges :

$$\rho_{tresh} = \nu d_{ROI}\,, \tag{3.12}$$

where ν is a multiplicative constant which can be set to be equal to the resolution of ρ in the Hough space. In this work, this second approach has been

[1]Note that the segments are represented in the image coordinate system, where the positive x axis points to the right, the positive y axis points downward and the positive z axis points toward the direction given by $\hat{\mathbf{z}} = \frac{\hat{\mathbf{x}}\wedge\hat{\mathbf{y}}}{\|\hat{\mathbf{x}}\wedge\hat{\mathbf{y}}\|}$.

preferred. Furthermore, it is imposed that the Euclidean distance between the farthest pair of endpoints of the two line segments must be less than d_{tresh} :

$$d_{p_{1a}-p_{2b}} \leqslant d_{tresh}\,, \tag{3.13}$$

where d_{tresh} is computed for each image as half of the mean length of the detected segments :

$$d_{tresh} = \frac{1}{2}\overline{(l_1, l_2, \ldots, l_n)}\,. \tag{3.14}$$

If the similarity conditions are met, then the two line segments are replaced with the line l_3, which is the line segment which joins the two farthest endpoints of l_1 and l_2 :

$$l_3\rho_3 = x\cos(\theta_3) + y\sin(\theta_3)\,. \tag{3.15}$$

The Hough parameters which defines l_3 can be computed as follows[2]. First, the angle between the positive x-axis and the joined segment is retrieved[3] :

$$\alpha = 90^\circ - \arctan\left(\frac{p_{1a,x} - p_{2b,x}}{p_{1a,y} - p_{2b,y}}\right), \tag{3.16}$$

from which it follows the value of θ_3

$$\theta_3 = \alpha - 90^\circ\,. \tag{3.17}$$

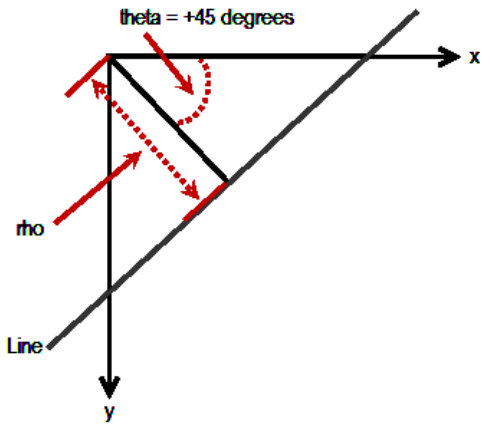


Figure 3.9: ρ and θ diagram

[2]Note that the computation is done in the image coordinate system previously defined

[3]Note that it is necessary to subtract 90° from the computed angle to take into account that in the image coordinate system negative the quadrant is the upper while the positive one is the lower

Now, to find ρ_3 it is sufficient to compute the intersection between the line which passes trough the origin and is also perpendicular to the joined segment :

$$m = \tan(\alpha) \tag{3.18}$$

$$x = \frac{m(p_{1a,x} - p_{1a,y})}{1/m + m}\,, \tag{3.19}$$

$$y = -\frac{1}{m}x\,, \tag{3.20}$$

$$\rho_3 = x\cos(\theta_3) + y\sin(\theta_3)\,. \tag{3.21}$$

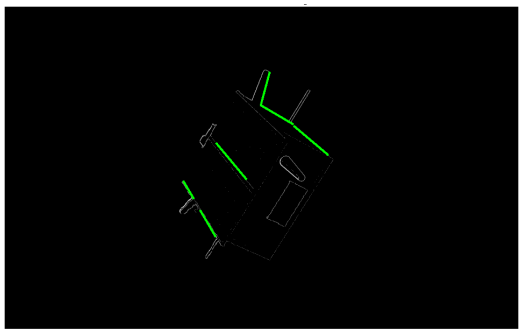
For what concerns the S&H stream instead, the edge function is applied by using a treshold of 0.4. After that, the Hough transform is applied, and a dedicated function removes all the detected lines which midpoints are located outside the ROI. In figures 3.10, 3.11, 3.12 are presented some preliminary results of the edge merging procedure applied to images produced using the toolbox presented in 2. The resolution of ρ and θ imposed to the Hough transform are set as $0.001d_{ROI}$ px, 5° over the range $[-90\ 89]$ with a maximum number of 5 peaks over a thresold of 0.1 for the WGE stream and $0.001d_{ROI}$ px, 0.1° over the range $[-90\ 89]$ with a maximum number of 10 peaks over a threshold of 0.1 for the S&H stream.

Merge Streams

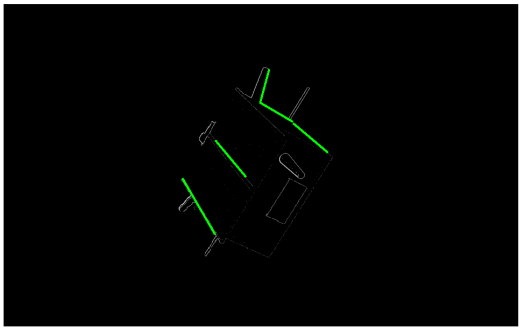
The final step of the feature detection procedure consist into merging the two streams of features obtained by using both the WGE and S&H. The aim of combining the two different streams is to identify different elements of the S/C silhouette. As stated in [60], the uniqueness check archived in this phase resolves the issue of detecting repeated edges as encountered in previous works. In contrast with what proposed in [60], which suggests to detect and merge pairs of close and similar line segments separately for the two streams and merge them after, here this job is accomplished in one step. The line segments of both streams are collected together and scanned by a loop which decides what to do with the current pair being analyzed on the basis of some geometrical conditions which are set. For example, pairs of close and similar line segments are detected and merged using conditions similar to the one imposed in section 3.1.2:

$$|\theta_1 - \theta_2| \leqslant \tilde{\theta}_{tresh}\,, \tag{3.22}$$

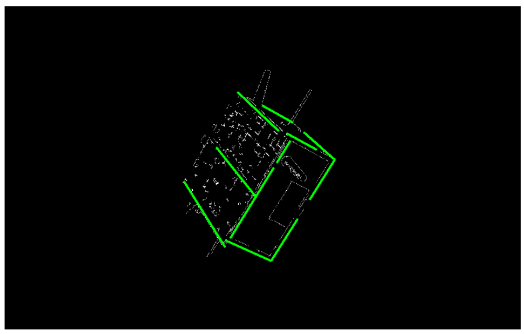
$$|\rho_1 - \rho_2| \leqslant \tilde{\rho}_{tresh}\,, \tag{3.23}$$



(a) WGE output

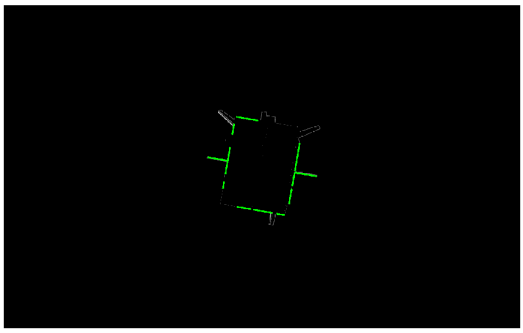


(b) WGE output after merging truncated edges

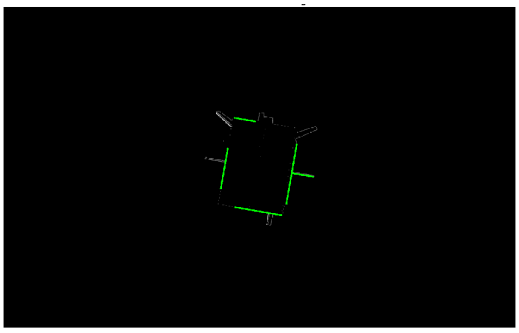


(c) S&H output

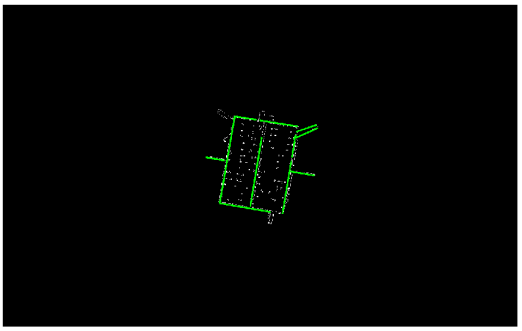
Figure 3.10: Results obtained applying edge detection and Hough transform on the two streams (1)



(a) WGE output

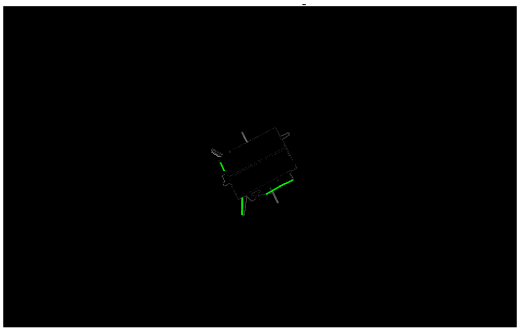


(b) WGE output after merging truncated edges

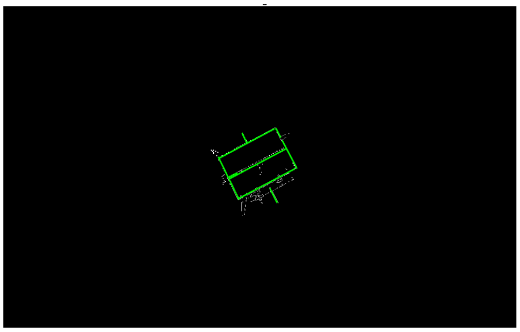


(c) S&H output

Figure 3.11: Results obtained applying edge detection and Hough transform on the two streams (2)



(a) WGE output



(b) S&H output

Figure 3.12: Case where no merging was required

where $\tilde{\rho}_{tresh}$ can be expressed as a function of the ROI diagonal lenght trough the multiplicative constant $\tilde{\nu}$ [25]:

$$\tilde{\rho}_{tresh} = \tilde{\nu} d_{ROI} \,. \tag{3.24}$$

Furthermore, it is imposed that the Euclidean distance between the midpoints must be less than half the length of the shorter line. With reference to figure 3.8 :

$$d_{p_{1b}-p_{2a}} \leqslant \tilde{d}_{tresh} \,, \tag{3.25}$$

where $\tilde{d}_{tresh}$ is computed for each pair being considered as half the length of the longer line segment :

$$\tilde{d}_{tresh} = \frac{1}{2} max(l_1, l_2) \,. \tag{3.26}$$

If the $\tilde{d}_{tresh}$ check is failed, the loop does two more checks before passing to the next pair. The first check controls if it is being considered a case were are present two long lines which intersect. The intersection can be computed by exploiting linear algebra. Considering two lines, l_1 (x_1, x_2,y_1,y_2) and l_2 (x_3, x_4,y_3,y_4) is possible to compute the following determinants :

$$dt_1 = det \left(\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_4 \\ y_1 & y_2 & y_4 \end{bmatrix} \right) \tag{3.27}$$

$$dt_2 = det \left(\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_3 & x_4 \\ y_1 & y_3 & y_4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ x_2 & x_3 & x_4 \\ y_2 & y_3 & y_4 \end{bmatrix} \right) \tag{3.28}$$

at this point, if $dt_1 \leqslant 0$ and $dt_2 \leqslant 0$ the two lines intersect, otherwise they do not. If they do intersect, then only the line segment composed by joining the farthest endpoints is retained if the following condition is met:

$$\epsilon = \frac{min(l_1, l_2)}{max(l_1, l_2)} > 0.25 \,. \tag{3.29}$$

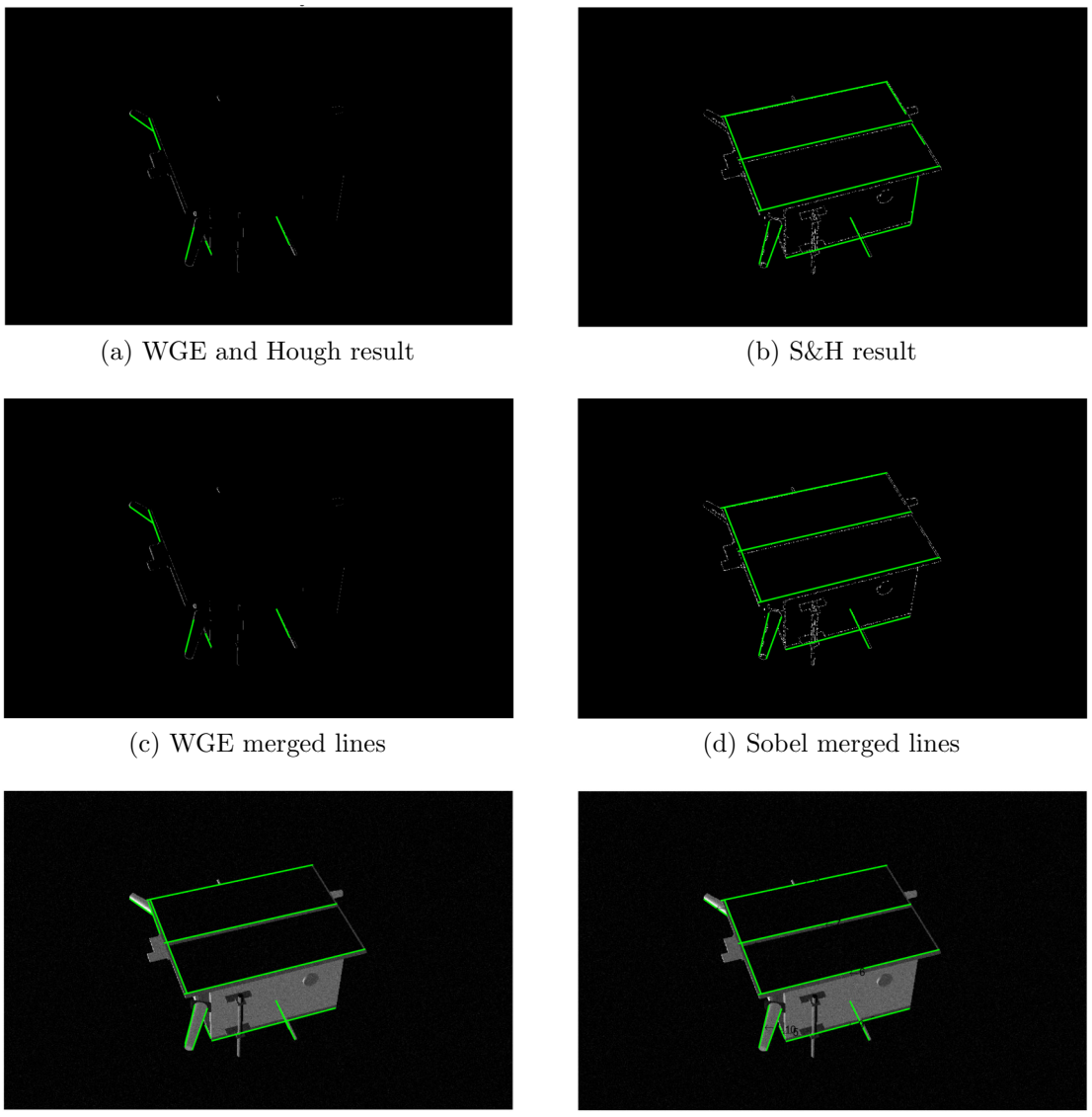
The second check instead controls if it is being considered a case were are present two long lines which are almost parallel. The parallelism is computed by exploiting the information about ρ :

$$\chi = 1 - \frac{min(\rho_1, \rho_2)}{max(\rho_1, \rho_2)} \tag{3.30}$$

if χ is less than 0.005 then the two lines are considered parallel and only the line segment composed by joining the farthest endpoints is retained if:

$$\epsilon = \frac{min(l_1, l_2)}{max(l_1, l_2)} > 0.25 \,. \tag{3.31}$$

Figures 3.13, 3.14, 3.15 shows the results of the merge streams procedure when applied to the images generated with the toolbox described in chapter 2.



(a) WGE and Hough result

(b) S&H result

(c) WGE merged lines

(d) Sobel merged lines

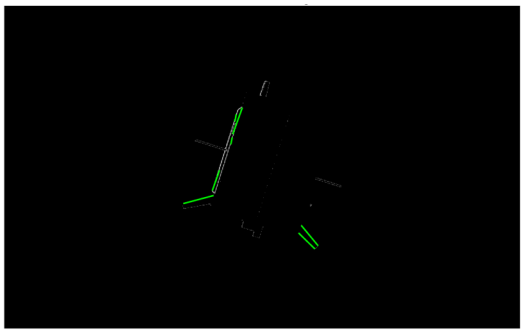
(e) Merge WGE and Sobel

(f) Final result

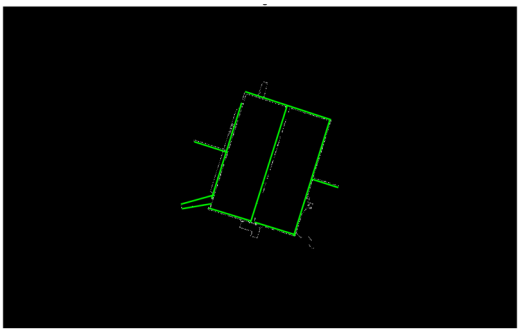
Figure 3.13: Results of the merging process (1)

Feature Synthesis and Perceptual Grouping

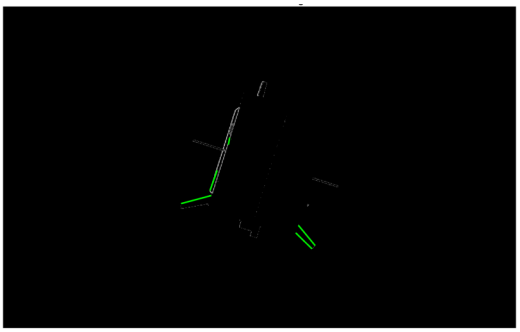
One of the innovation introduced in the SVD algorithm is what by the original authors has been defined as "Feature Synthesys", which allows to organize the simple segments output from the merge of the WGE and S&H streams into into high-level geometrical groups called "Perceptual Groups" in order to reduce



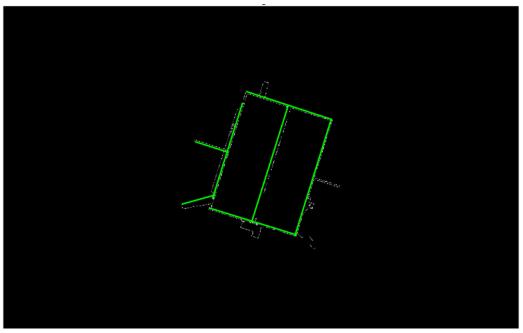
(a) WGE and Hough result



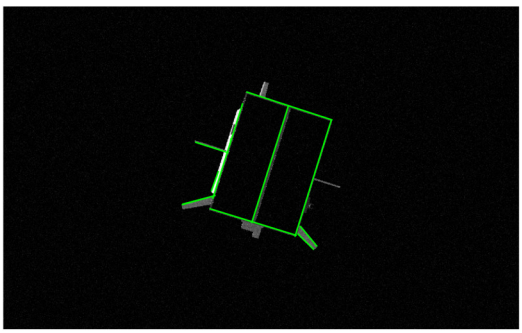
(b) S&H result



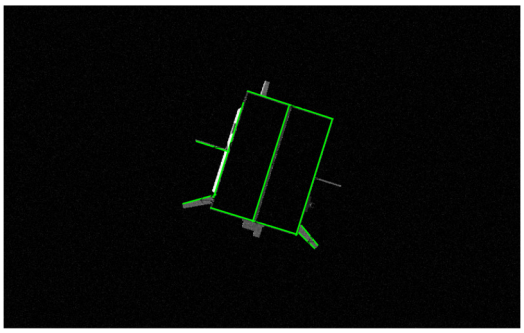
(c) WGE merged lines



(d) Sobel merged lines

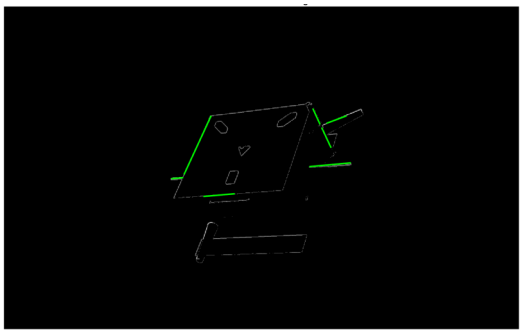


(e) Merge WGE and Sobel

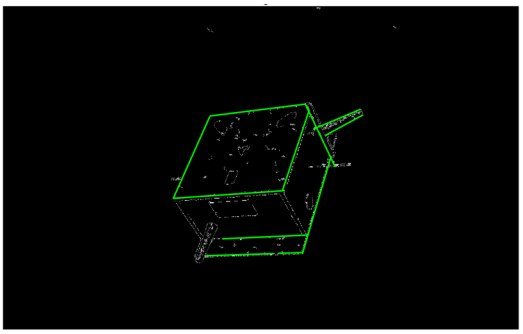


(f) Final result

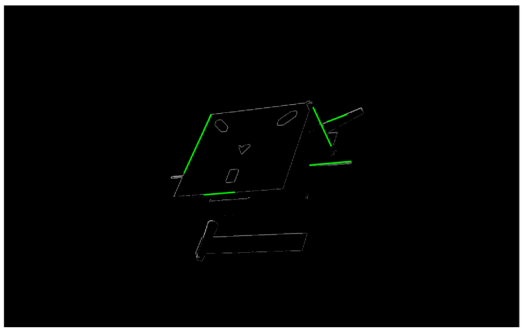
Figure 3.14: Results of the merging process (2)



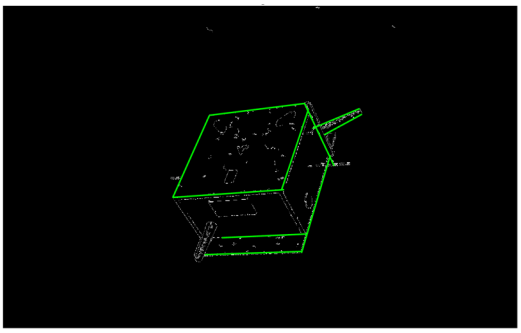
(a) WGE and Hough result



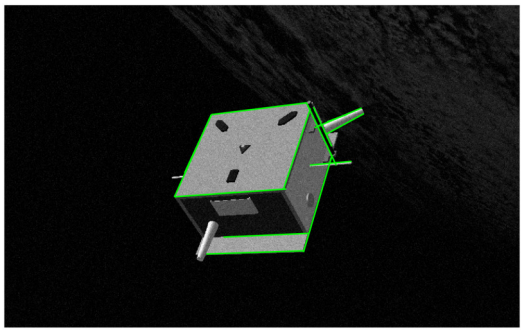
(b) S&H result



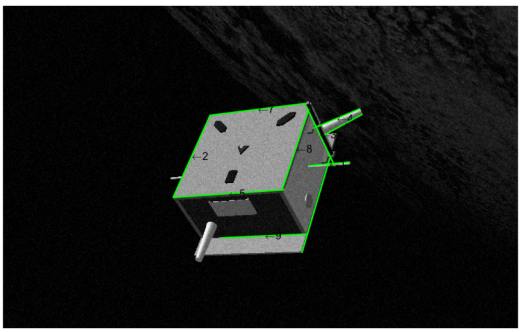
(c) WGE merged lines



(d) Sobel merged lines



(e) Merge WGE and Sobel



(f) Final result

Figure 3.15: Results of the merging process (3)

the search space of the correspondence problem. Being able to correctly define perceptual groups is of crucial importance to have a successful pose initialization. As stated in [24], to uniquely solve the P-n-P problem are required at least six correspondences between model and image points. The number of possible correspondences, so, the number of hypothetical pose solution given n points in the image and m points in the 3-D model therefore can be expressed as [60]:

$$\binom{m}{6}\binom{n}{6}6!\,.$$

The idea presented by Sharma *et al.* so is to reduce the solutions' search space by just using a small set of high level feature groups instead of using a large number of feature point. For example, it is sufficiently safe to say that four image points may belong to a polygonal feature such as a solar panel, then a unique solution can be found by just using four correspondences. The feature synthesis implementation proposed by Sharma *et al.* groups the features into five high-level perceptual groups, which, in order of increasing complexity are :

- parallel pairs;
- proximity pairs;
- parallel triads;
- proximity triads;
- closed polygonal tetrads;

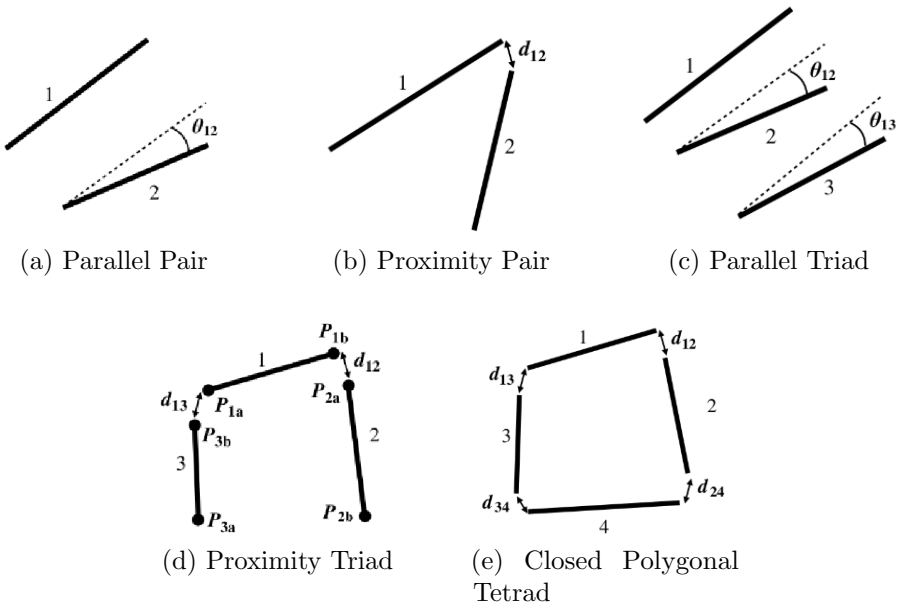
moreover, antennas are treated as a separate feature group. As done for the merging procedure, the perceptual groups are recognized by examining several geometrical constraints.

Antennas are detected by selecting among all line segments only the one for which the following condition is met :

$$l_1 \leqslant \tau d_{ROI}\,, \tag{3.32}$$

where τ is a multiplicative constant, which [60] suggest to take equal to $1/3$. Before checking if a line can be inserted into an high level feature group, is always checked if it cannot be characterized as antenna. With reference to figure 3.16a, parallelism is enforced by checking the the angular difference between the two line segments:

$$\theta_{12} = |\theta_1 - \theta_2| \leqslant \theta_{max}\,, \tag{3.33}$$



(a) Parallel Pair (b) Proximity Pair (c) Parallel Triad

(d) Proximity Triad (e) Closed Polygonal Tetrad

Figure 3.16: High-level feature groups [60]

However, as suggested in [25], to avoid detecting spurious parallel pairs it is useful to introduce two more geometric constraints, which are the minimum radial distance and the minimum lines' length ratio :

$$\rho_{12} = |\rho_1 - \rho_2| \geq \rho_{min} \,, \tag{3.34}$$

$$\epsilon_{12} = \frac{min(l_1, l_2)}{max(l_1, l_2)} \geq \epsilon_{min} \,. \tag{3.35}$$

Proximity pairs instead are detected by checking the distance between their closest endpoints. With reference to figure 3.16b, the geometric condition checked is:

$$d_{12} \leqslant d_{max} \,, \tag{3.36}$$

where d_{max} can be adaptively computed in function of the ROI diagonal length trough a multiplicative constant:

$$d_{max} = \xi d_{ROI} \,. \tag{3.37}$$

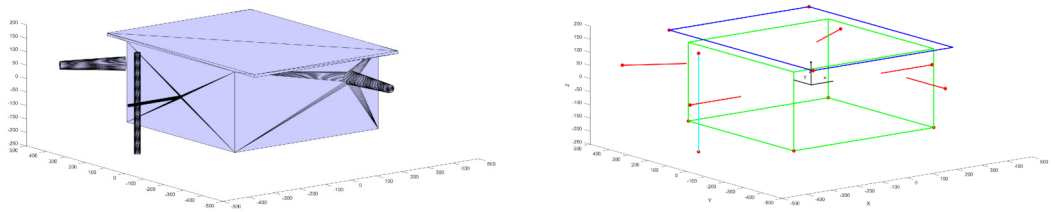
Furthermore, in order to prevent duplicated edges to be grouped as proximal pair, in [25] is proposed a supplementary constraint on the minimum angular difference between the two line segments being considered:

$$\theta_{12} = |\theta_1 - \theta_2| \geq \theta_{min} \,. \tag{3.38}$$

Parallel triads can be detected by selecting among all parallel pairs, the ones which shares a line segment. Similarly, proximity triads are detected by scanning all the proximity pairs and extracting the ones which shares a line segment and satisfy the following geometric condition:

$$(P_{1a} - P_{3a})(P_{1b} - P_{3b}) > 0\,. \tag{3.39}$$

Finally, polygonal tetrads that have two segments in common are classified as closed polygonal tetrads. Similarly, the perceptual grouping is applied to a reduced CAD model of the target S/C. As proposed in [60], the CAD model introduced in chapter 2 is reduced to only contain a low number of features. The number of features present in the wireframe reduced modules must be tuned in order to reduce all possible pose ambiguities and to reduce the number of feature correspondence hypotheses to speed up the pose determination process. The origin of the body frame is located in correspondence of the CG of the S/C.



(a) Original CAD model (b) Reduced CAD model

Figure 3.17: Full CAD model and reduced one

Once the CAD model is input in MATLAB[4] and reduced, the same feature groups can be extracted from the 3-D model by applying 3-D geometry definitions instead of 2-D ones. For example, parallelism can be defined using the unit vectors constructed using end-points of the lines. Similarly, proximal pairs can be defined based on the smallest possible distance between the end points of the line segments, etc. From the 3-D model shown in figure 3.17 the following high level features have been detected :

- 18 parallel pairs;
- 16 proximity pairs;
- 12 parallel triads;
- 11 proximity triads;
- 2 closed polygonal tetrads;

[4]The following function can be used: `https://www.mathworks.com/matlabcentral/fileexchange/30923-fast-stl-import-function`

in addition to the five antennas and the lateral bar which have been placed on the S/C.

3.1.3 Model Matching

Once all feature groups are extracted from the 3-D model and the 2-D image, the baseline idea is to hypothesize correspondences between the endpoints of the 3-D model and 2-D line segments for each matching feature group pair through simple combinations. The point correspondences then are stored in a match matrix, which can be then input to the desired P-n-P solver.The feature groups can ranked according their geometrical complexity, and only the most geometrically complex feature group detected in the image are considered.

Feature group	Number of points per feature group	Number of feature groups in image	Number of feature groups in 3-D model	Number of rows in match matrix
Closed polygonal tetrad	4	ϕ_a	ϕ'_a	$8\phi_a\phi'_a\phi_f\phi'_f$
Open polygonal tetrad	4	ϕ_b	ϕ'_b	$8\phi_b\phi'_b\phi_f\phi'_f$
Parallel triad	6	ϕ_c	ϕ'_c	$24\phi_c\phi'_c$
Parallel pair	4	ϕ_d	ϕ'_d	$8\phi_d\phi'_d\phi_f\phi'_f$
Proximal pair	3	ϕ_e	ϕ'_e	$2\phi_e\phi'_e\phi_f\phi'_f$
Antenna	1	ϕ_f	ϕ'_f	——

Table 3.1: Expected number of rows in the match matrix (column 5) based on the most geometrically complex feature group detected in the image (column 1) [60]

In table 3.1 is shown the number of rows the match matrix should have for a spacecraft like the tango S/C in a typical scenario were at lest three antennas are detected. Unlike previous view based pose estimation techniques, in the SVD architecture not all possible feature matches are treated identically. Instead, a small set of matches is hypothesized and then verified, by combining most complex feature group with simplest feature groups detected in order to have enough point matches to fed to the pose solver. In particular, for S/Cs like Tango, Sharma *et al.* propose to to combine point correspondence from complex feature groups with point correspondences from the antennas feature group, since the S/C has several antennas which are visible from most viewing angles.

3.1.4 Pose Determination

For the pose determination step the original authors propose to couple the eP-n-P pose solver [38] to each combination of feature correspondence present in the match matrix to archive the five best solutions and then use those for pose refinement by using the NR method to solve equations (3.1) and (3.2) using each pose solution

as initial guess in order to minimize the following fit error:

$$\mathbf{E_i} = \left[u_i - \left(\frac{x_C}{z_C} f_x + C_x \right), \left(v_i - \frac{y_C}{z_C} f_y + C_y \right) \right] \text{ for } i = 1, ..., n\,. \tag{3.40}$$

In this work however it has been opted to use a built-in MATLAB function which returns orientation and position of a calibrate camera by means of a P3P solver [26] coupled with a RANSAC algorithm for outlier rejection [63] in a coordinate system which is the one of the map input to the function (so to speak, the wireframe 3-D model previously introduced). The P3P solver produces up to four symmetrical solutions using three points. Less solution can be produced if some geometrical and algebraic conditions are met. Since the P-n-P is prone to error if any outliers are present in the set of give point correspondences, the RANSAC is used to make the final solution more robust.

3.2 Conclusions

This chapter is concluded showing to the reader some cases of successful pose initialization when using the SVD architecture which has been described and analyzed trough the whole chapter. In the table 3.2 the interested reader can see the values of some of the ROI parameters which have been described in the trough the previous sections used to tune the SVD algorithm in order to analyze the image produced with the toolbox presented in chapter 2.

Symbol	Value
κ_1	0.031
κ_2	0.0085
κ_3	0.086
κ_4	0.0146
η	0.03
θ_{tresh}	10°
ν	0.1
$\tilde{\theta}_{tresh}$	10°
$\tilde{\nu}$	0.129
θ_{max}	10°
ρ_{min}	$0.05 d_{ROI}$
ϵ_{min}	0.7
ξ	0.03
θ_{min}	10°

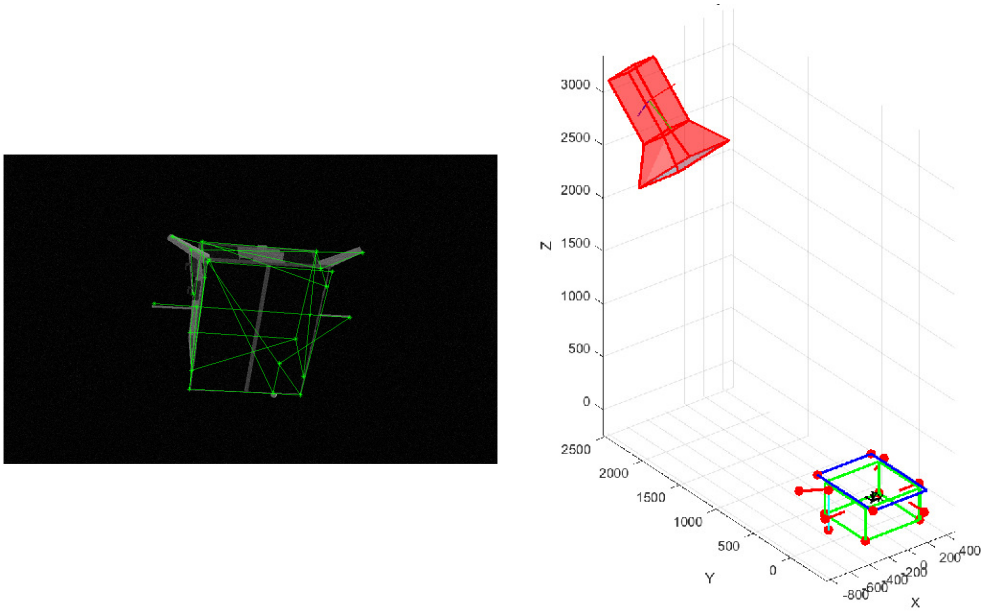
Table 3.2: Parameters used to tune the SVD algorithm

The accuracy of the estimated pose can be evaluated by defining the following errors:

$$\mathbf{E_t} = |\mathbf{t_c}^{true} - \mathbf{t_c}^{est}| \tag{3.41}$$

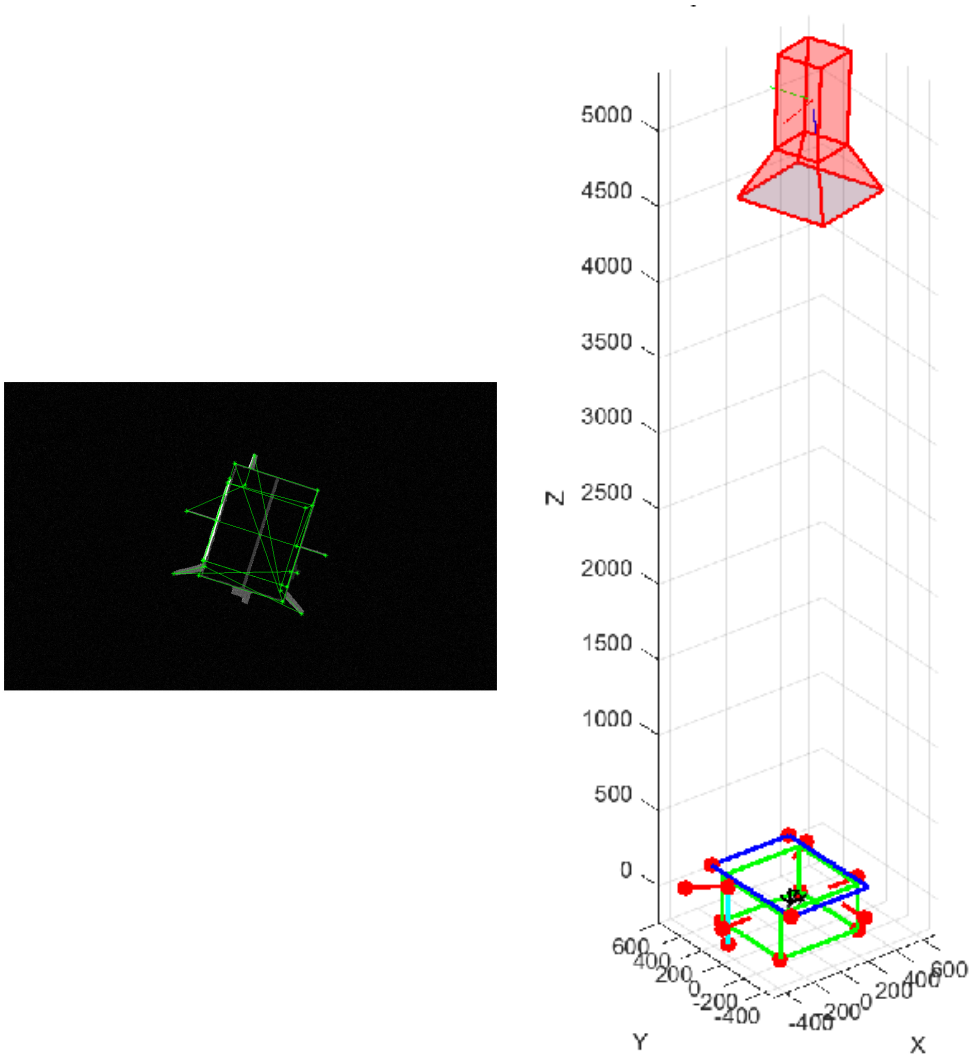
$$\mathbf{A_{diff}} = \mathbf{A_{TC}}^{est}(\mathbf{A_{TC}}^{true})^T \tag{3.42}$$

where $\mathbf{E_t}$ represents the absolute difference between the ground truth position of the S/C (the one imposed at the moment when the image was generated) and the estimated position provided by the pose solution, and $\mathbf{A_{diff}}$ is the DCM representing the relative rotation between the ground truth value (the one imposed at the moment when the image was generated) and the estimated value of $\mathbf{A_{TC}}$.



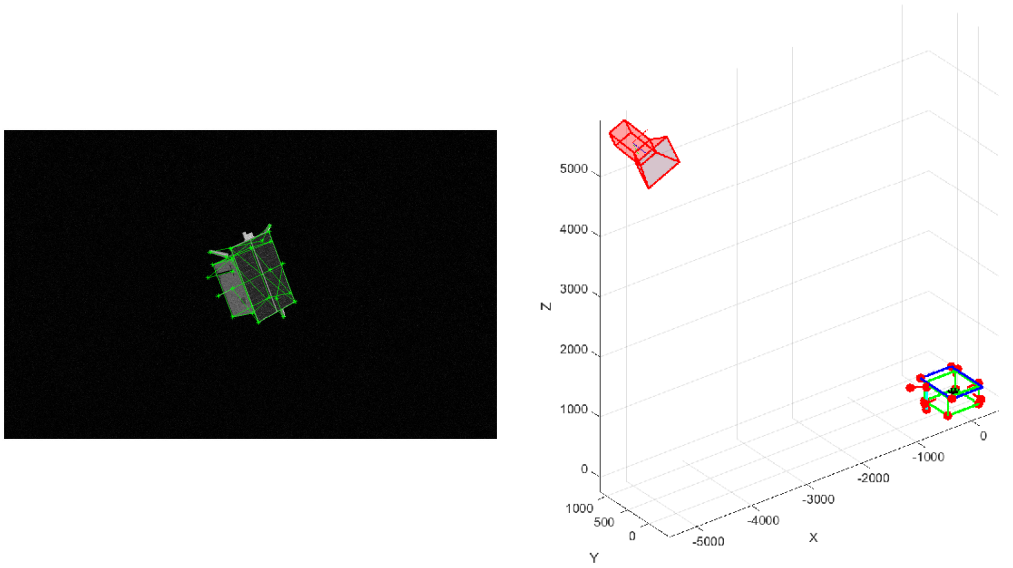
(a) Map superposed to the image (b) 3-D camera pose in map coordinates

Figure 3.18: Case of successful pose initialization (1)



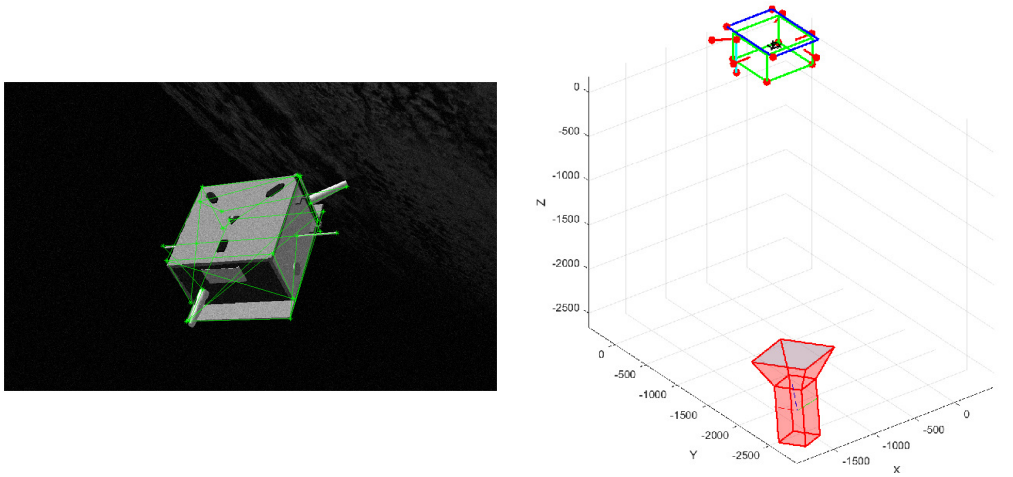
(a) Map superposed to the image (b) 3-D camera pose in map coordinates

Figure 3.19: Case of successful pose initialization (2)



(a) Map superposed to the image (b) 3-D camera pose in map coordinates

Figure 3.20: Case of successful pose initialization (3)



(a) Map superposed to the image (b) 3-D camera pose in map coordinates

Figure 3.21: Case of successful pose initialization (4)

Chapter 4

Results

"Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program."

Linus Torvalds

In this chapter the reader will first be presented with a qualitative analysis of the images produced with the toolbox illustrated in chapter 2. After that, the preliminary validation of the SVD algorithm presented in chapter 3 when applied to the same images will be discussed.

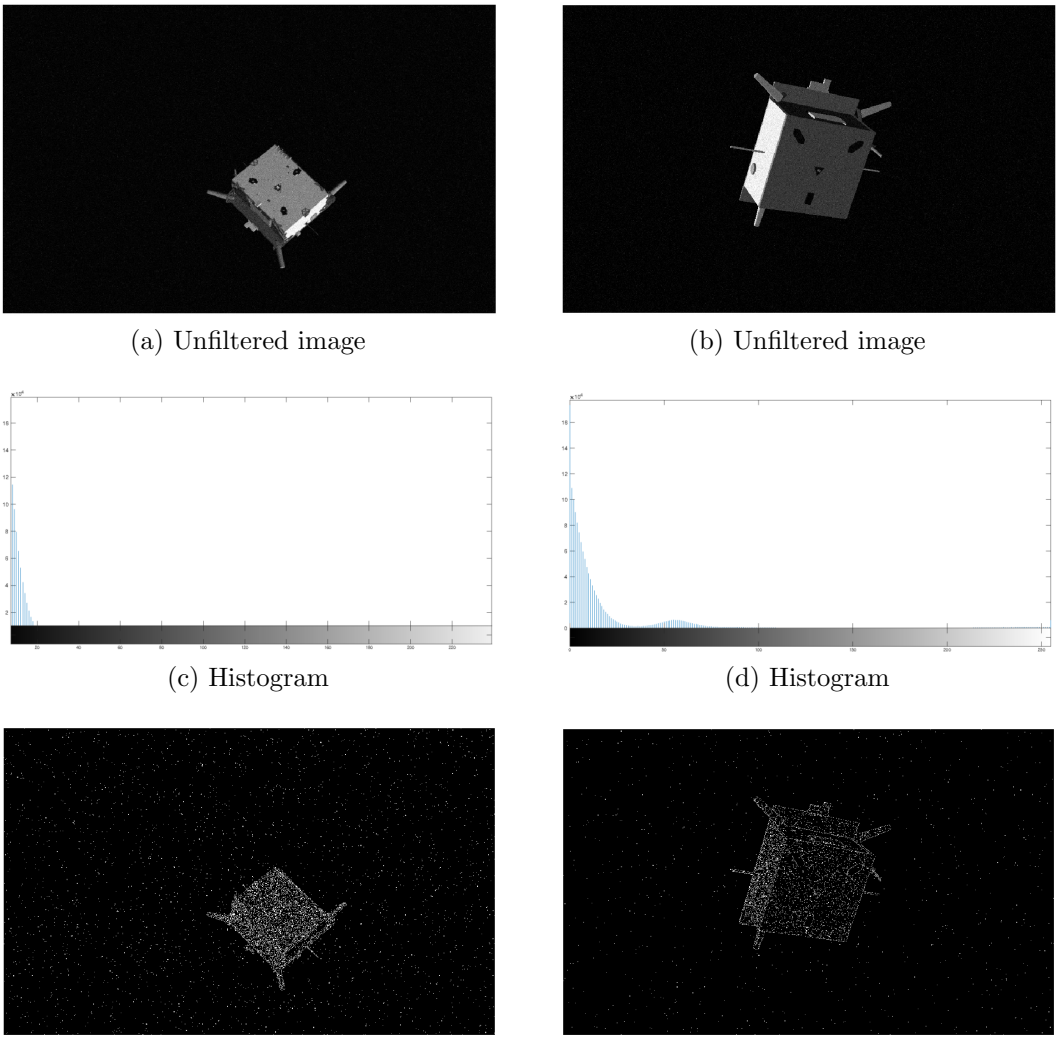
4.1 Qualitative Assessment

The aim of the toolbox presented in chapter 2 was to create batches of realistic images of a given CAD of a S/C in order to cope with the lack of an actual facility to produce images from a scale 3-D printed model of the target itself, as done in [6]. Validating the goodness of the generated images is extremely difficult because of the actual lack of high-resolution and high-quality images taken in orbit by a real camera of a real S/C. So, the more realistic intent of this evaluation is therefore to compare the generated images to the images found in the SPEED data-set [1], in order to identify similarities and differences. As proposed in [33], the images can be compared in three ways:

- visual comparison;
- image histogram comparison;
- basic feature extraction using a Sobel edge detector.

[1]The SPEED data-set is freely available at `https://kelvins.esa.int/satellite-pose-estimation-challenge/`

Unfortunately, the orbit and the true pose but more importantly the illumination conditions of each of the images contained in the SPEED data-set was not available at the time of writing this thesis, so it is impossible to present to the reader a one-to-one comparison. Still, it is worth to compare images which can be reasonably considered similar.



(a) Unfiltered image

(b) Unfiltered image

(c) Histogram

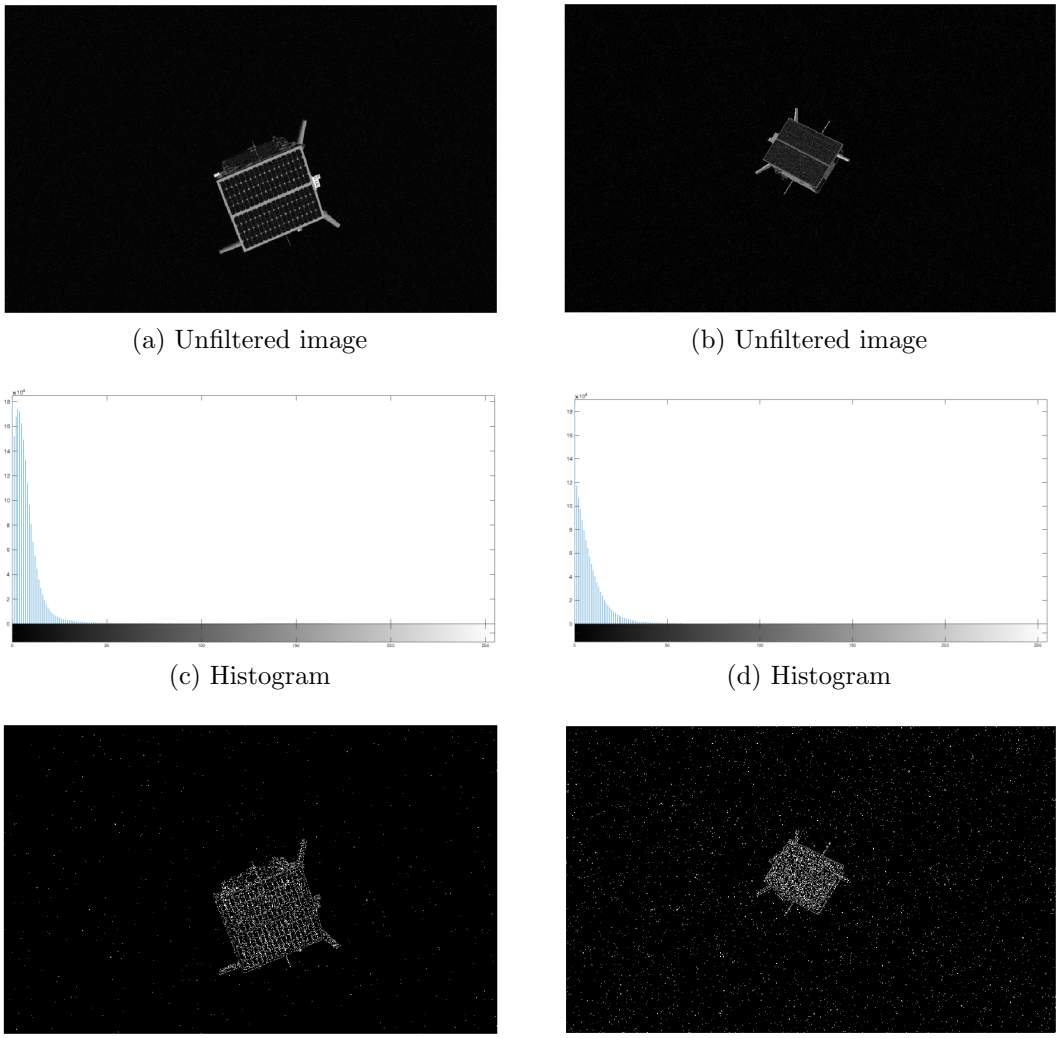
(d) Histogram

(e) Sobel image

(f) Sobel image

Figure 4.1: Comparison between the SPEED data-set (left) and images generated using the toolbox presented in chapter 2 (right), 1

In figures 4.1, 4.2, 4.3 are presented three simulated images of the Tango S/C on a black background. Overall, it is possible to recognize that the images belonging to the SPEED shows a better representation of the external look of the S/C, for both the solar panels and the external appendages and external structures. Usually, for rendering images of S/Cs, the manufacturer makes available textures



(a) Unfiltered image

(b) Unfiltered image

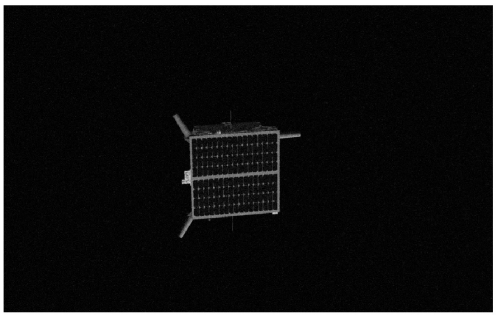
(c) Histogram

(d) Histogram

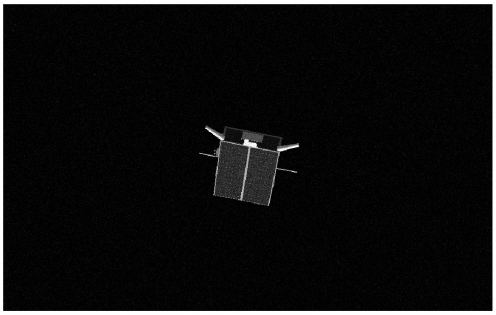
(e) Sobel image

(f) Sobel image

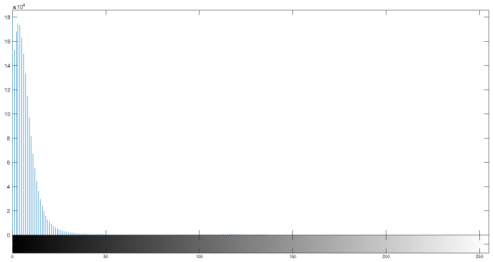
Figure 4.2: Comparison between the SPEED data-set (left) and images generated using the toolbox presented in chapter 2 (right), 1



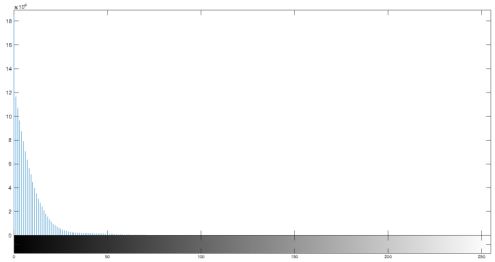
(a) Unfiltered image



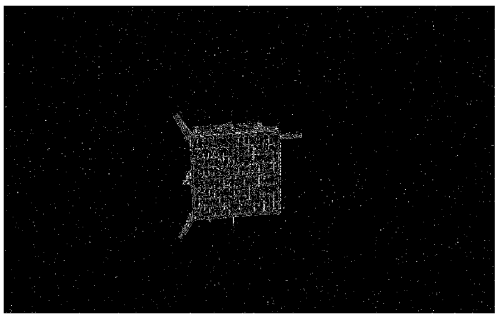
(b) Unfiltered image



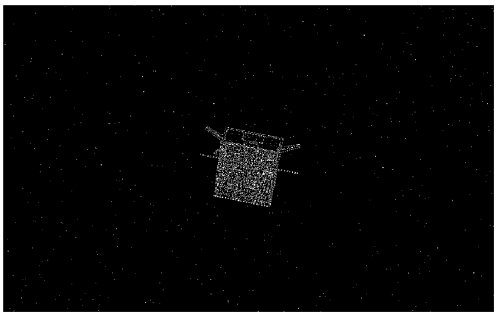
(c) Histogram



(d) Histogram

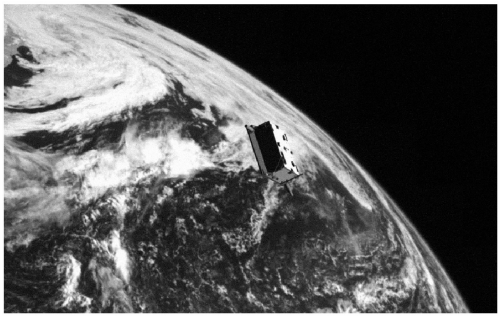


(e) Sobel image

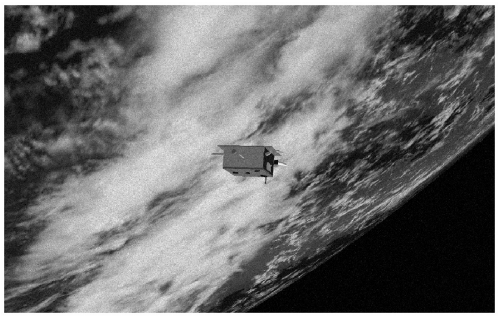


(f) Sobel image

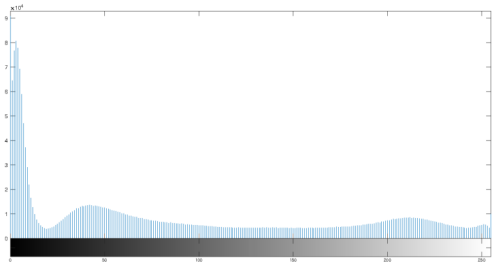
Figure 4.3: Comparison between the SPEED data-set (left) and images generated using the toolbox presented in chapter 2 (right), 1



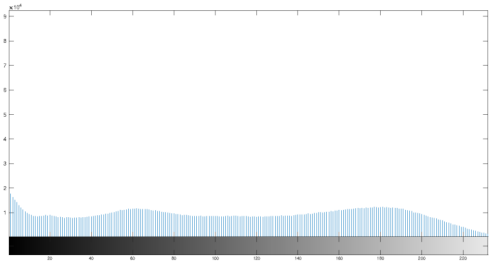
(a) Unfiltered image



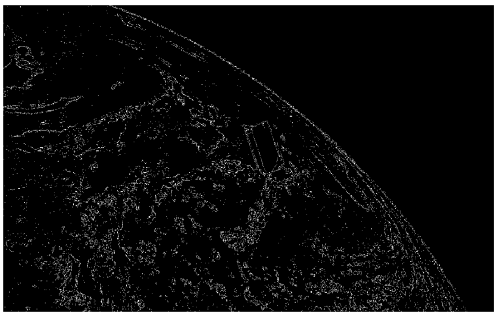
(b) Unfiltered image



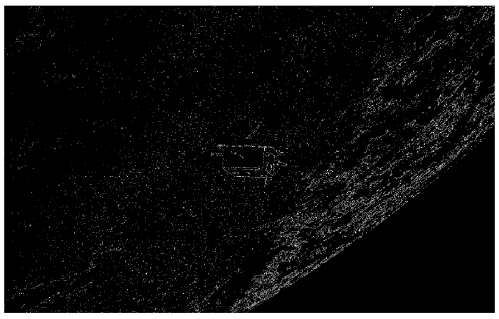
(c) Histogram



(d) Histogram



(e) Sobel image



(f) Sobel image

Figure 4.4: Comparison between the SPEED data-set (left) and images generated using the toolbox presented in chapter 2 (right), 1

of the external look of the whole S/C. For this project unfortunately, due to the unavailability of said textures, the whole look of had to be recreated by hand, doing several experiments and comparisons. This is particularly visible when looking at solar panels, where the SPEED images are of an order of magnitude superior in terms of accuracy and realness of what is being represented. Despite said issues however, by looking at the forms of the histograms it is possible to recognize that histograms belonging to the SPEED images and histograms belonging to the images used though this project are essentially similar. They both shows the same exponential behavior. The images belonging to the SPEED data-set shows an overall greater concentration of dark values with respect to the ones generated with the toolbox presented in chapter 2. This is likely to be due to slightly different scenes being represented in terms of illumination conditions and shadows. The Sobel edge images are also very similar. The images generated for this project presents a slightly higher level of noise (recognizable by the more presence of little white pixels). This can be due to a different σ^2 value used to add Gaussian white noise to the image during the post-processing phase. Lastly, in figure 4.4 is showed a comparison were the Earth is in the background. As it can be clearly seen by the reader, the Earth is represented more accurately in the SPEED data-set, but this is simply due to the fact that for the SPEED data-set the Earth images comes from actually real space imagery. As described in [59], the Earth images used for the SPEED data-set are composed by 72 actual images of the Earth captured by the Himawari-8 geostationary meteorological satellite. The 72 images each provide a 100×10^6 pixels resolution disk-view of the Earth and were taken 10 minutes apart from each other over a period of 12 hours. The quality the Earth has in the images generated with the toolbox developed during this work can be enhanced by using a texture with an higher resolution, however this comes at the cost of long rendering times and greater RAM usage for rendering the single image. Generating one single image where the Earth is present in the background takes approximately 3 s and occupy *circa* 2 Gb of RAM on an Intel Core i7-4500U CPU when using mercator images rescaled to 21600×10800 pixel resolution. Better results can be archived using the full 43200×21600 pixel original mercator images but rendering times as well as RAM usage would grow too much for the generation of a data-set of hundreds of images on the previously mentioned H/W.

4.2 SVD Architecture Tests

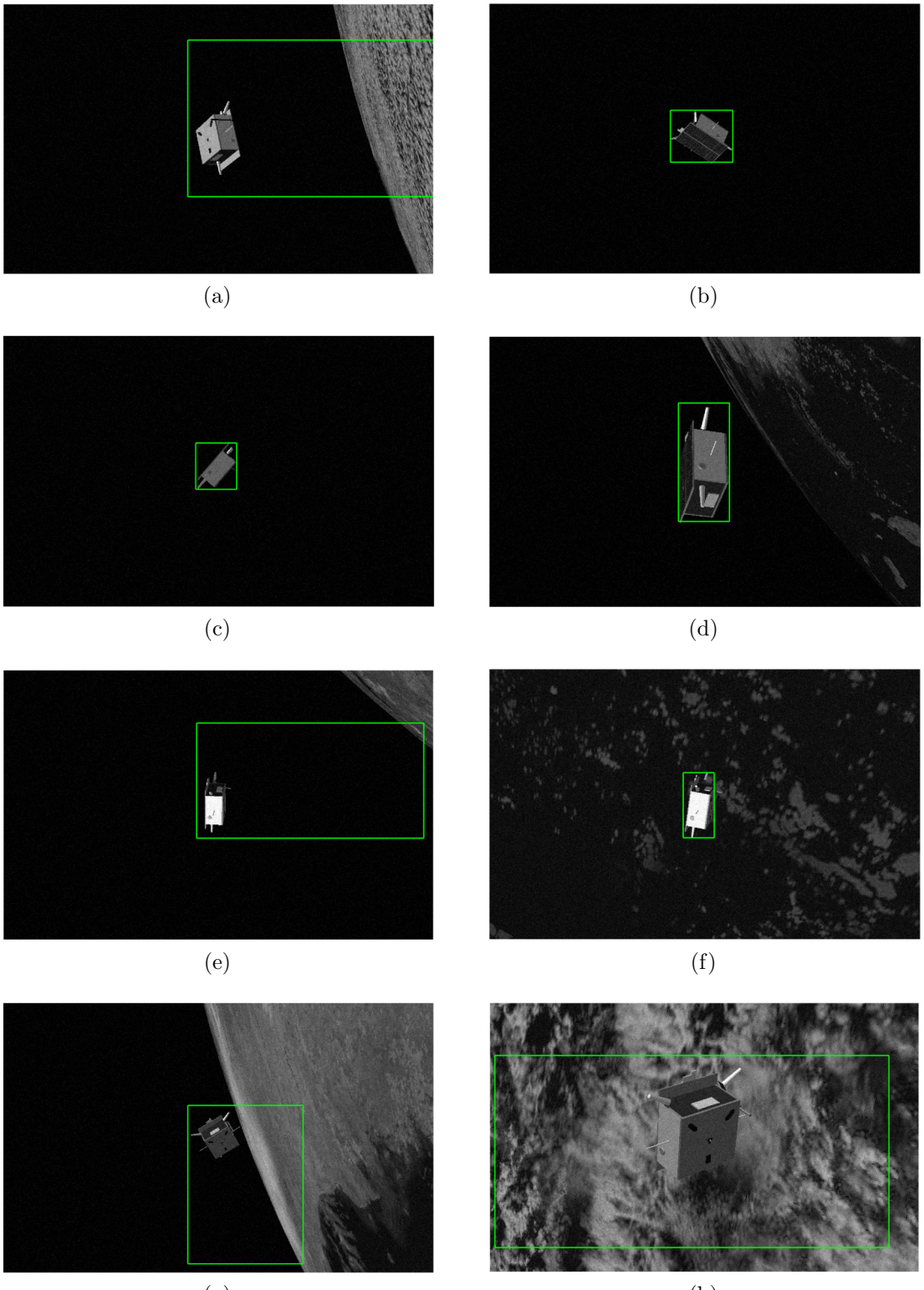
A further validation of the generated data-set can be obtained by analyzing the synthetic images using a CV algorithm. In fact, if the quality of the generated images is poor, where for poor it is intended not being photorealistic, the CV algorithm will not work consistently. Among all state-of-the-art CV algorithms available, the SVD algorithm has been selected because of its simplicity and effectiveness. The identification of a correct ROI is of vital importance since the

edge detection process as well as the merging edges block depends on geometrical parameters which are set as multiplicative constants of the diagonal length of the ROI. From a first batch of tests realized using the images generated using the toolbox presented in chapter 2, the ROI detection performances of the WGE technique are consistent with what found in [60] and [25]. When the image has a black or nearly black background, the results of the WGE are very good, and in almost all cases it's capable of identifying the correct ROI. However, the presence of a composite background (such as the Earth) in some cases negatively affects the performances of the WGE technique. This is due to the fact that the presence of the Earth produces spurious element in the gradient image, which are not eliminated by the filtering procedure. This particular behavior can be observed for example in figures 4.5a, 4.5h and 4.6e.

Moreover, the failure in the detection of the ROI, also also involves the impossibility of rejecting spurious lines which do not belong to the silhouette of the S/C (as can be observed in figures 4.10, 4.11 and 4.12), and which would have been rejected in the case of a correct ROI selection.

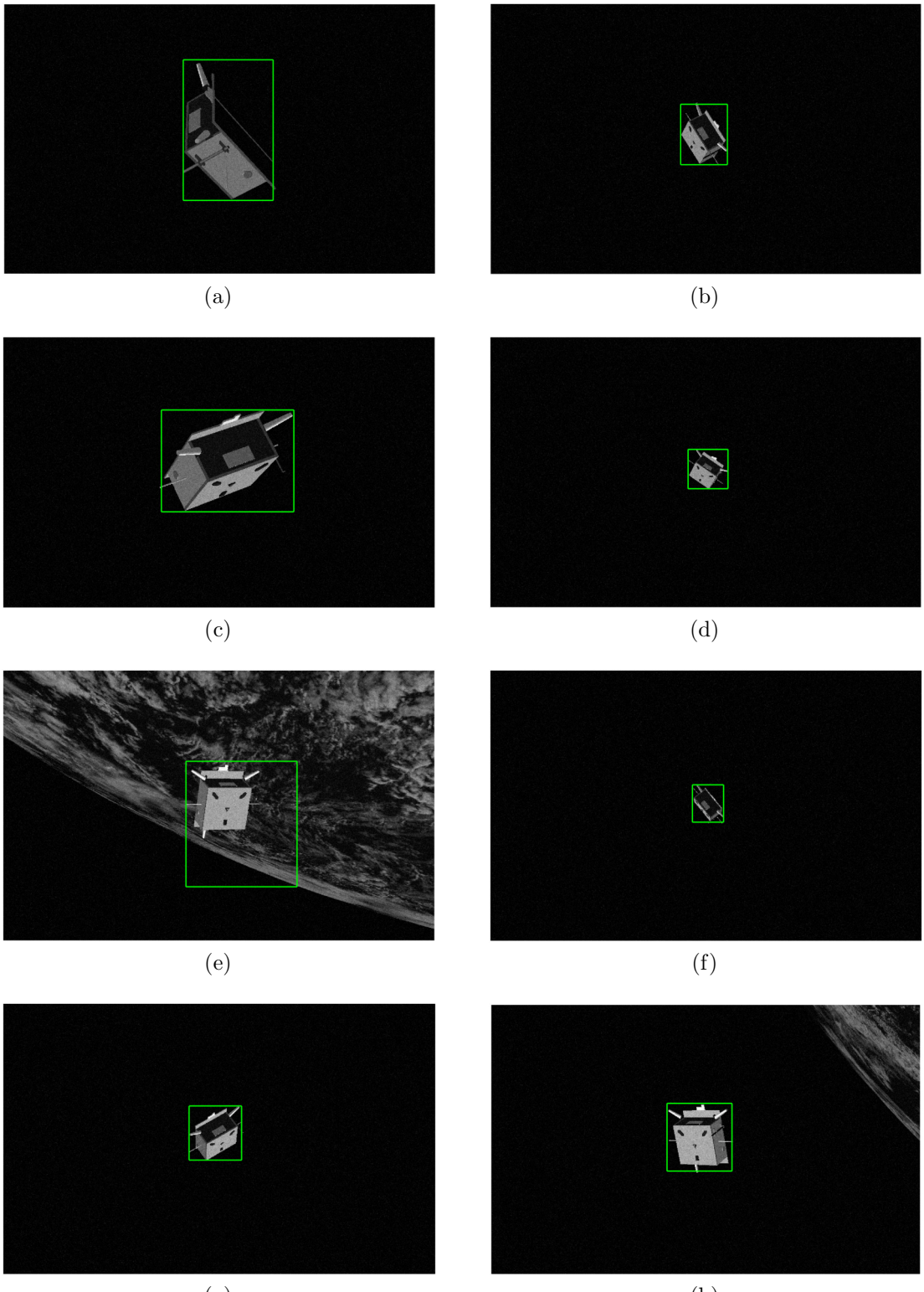
On one hand, one could think to increase more the filtering depth of the image, but on the other hand this will penalize too much the analysis of images which do not have a composite background. As previously explained, a failure into the detection of the ROI greatly affects the performances of the SVD algorithm, since all the quantities which are computed as multiplicative constants of the ROI diagonal length will be biased. For what concerns the edge detection procedure instead, as observed in [60], further improvements must be made because it is still susceptible to producing spurious edges. Despite this being bad from the point of view of the CV algorithm, this is good for what concerns the goodness of the generated data-set, which can reproduce more or less the same kinds of issues which the SPEED data-set had. In particular, it has been observed during this work that the most critical images are the ones where the solar panels are clearly visible and when the target S/C is far from the camera. For what concerns the former case, it is particularly evident from figure 4.13b that the culprit is likely due to the Sobel edge detector.

While all the spurious points are filtered during by the WGE stream, they are not in the Sobel stream. Subsequently, when applying the Hough transform some spurious edge are detected. During the work on this project, some efforts have been made to cope with this issue which however has not been completely resolved, from what can bee seen in figure 4.13c. On one hand, more treshold could be inserted into the image processing subsystem to take into account the presence of short spurious line and and remove them. On the other hand however, is really hard to fine tune those treshold in order to not remove other short edges which instead are useful, such as antennas. There are also some particular attitudes where the Hough transform fails on the WGE filtered image, like for example what can be observed from figure 4.14. In that case the ground truth relative pose was



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 4.5: ROI detection tests



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 4.6: ROI detection tests

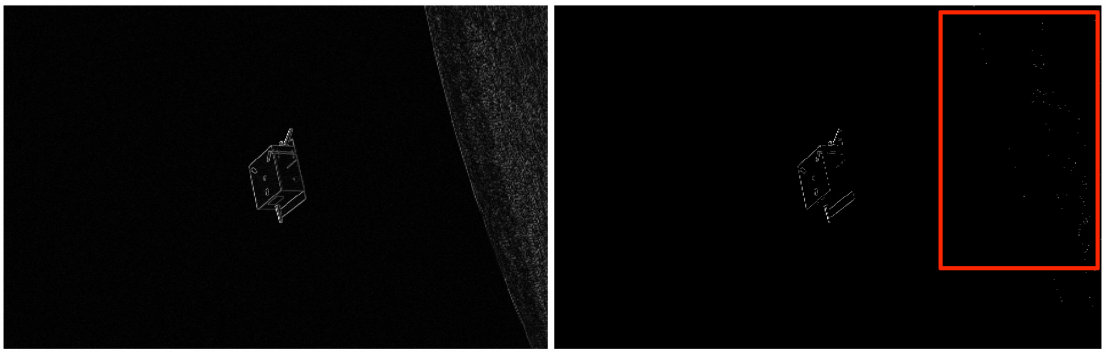


Figure 4.7: Gradient image of figure 4.5a. Left is normalized gradient, right is normalized gradient after tresholding.

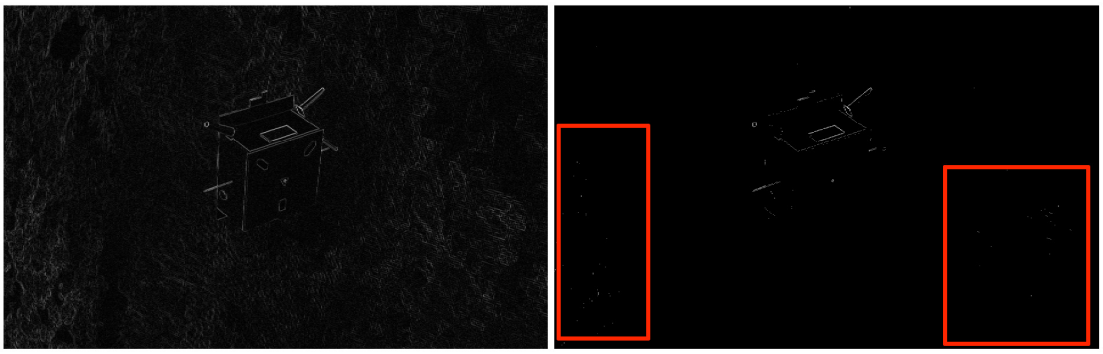


Figure 4.8: Gradient image of figure 4.5h. Left is normalized gradient, right is normalized gradient after tresholding.

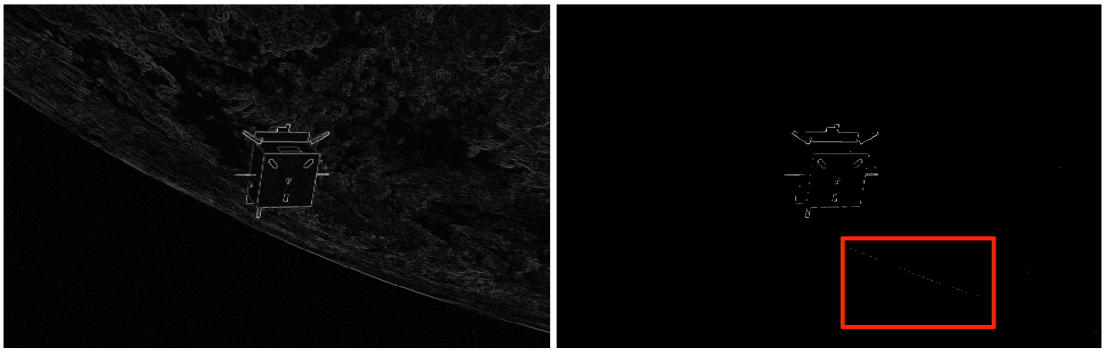


Figure 4.9: Gradient image of figure 4.6e. Left is normalized gradient, right is normalized gradient after tresholding.

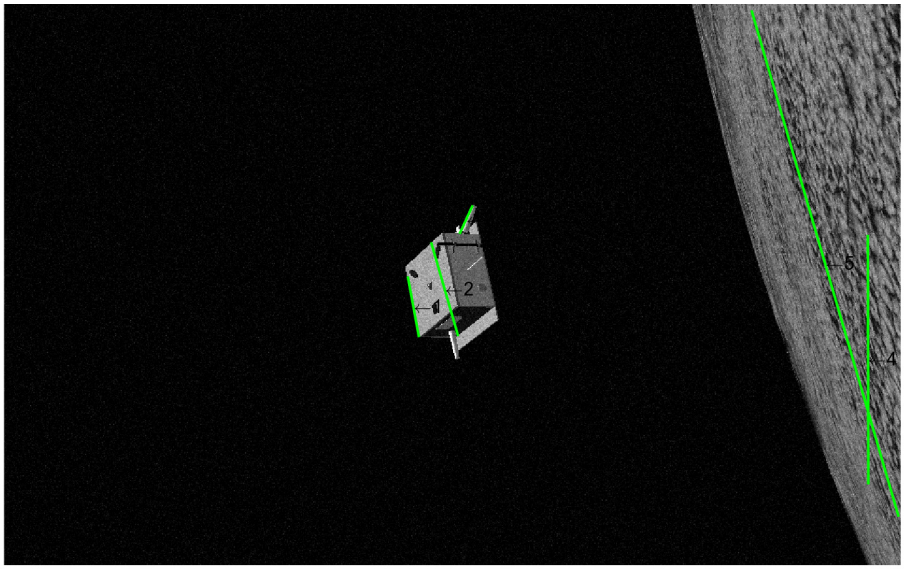


Figure 4.10: Edge detection on figure 4.5a

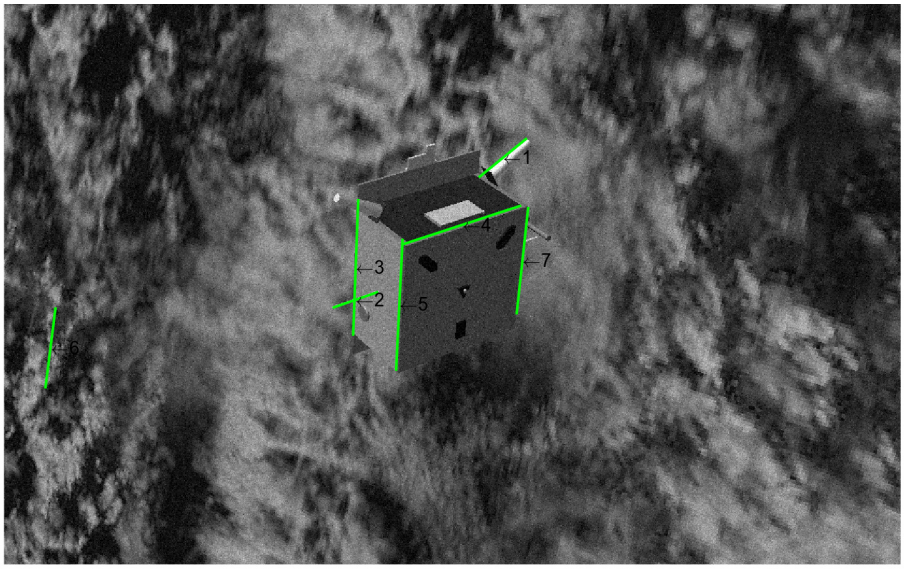


Figure 4.11: Edge detection on figure 4.5h

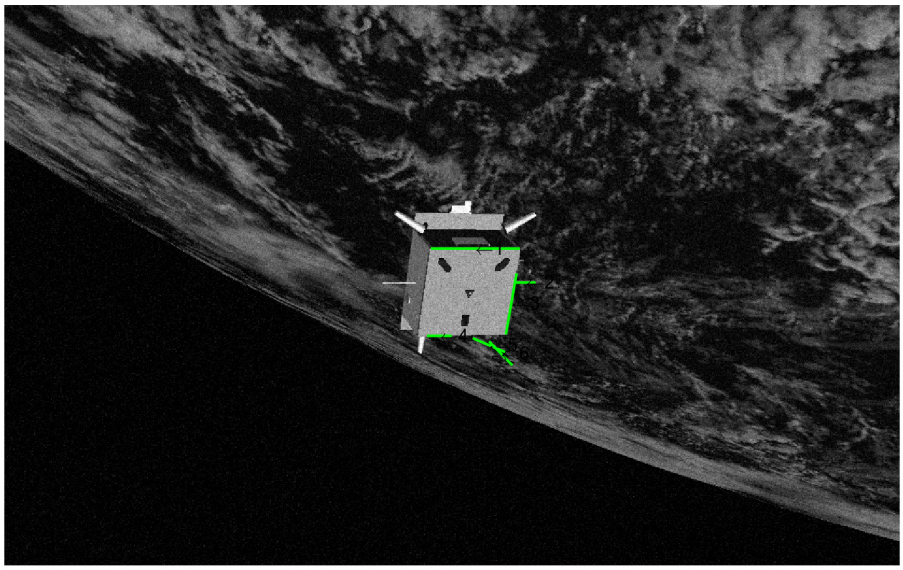
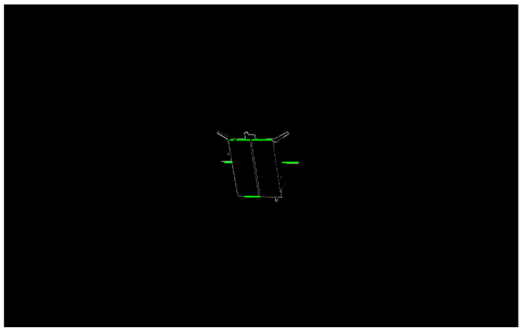
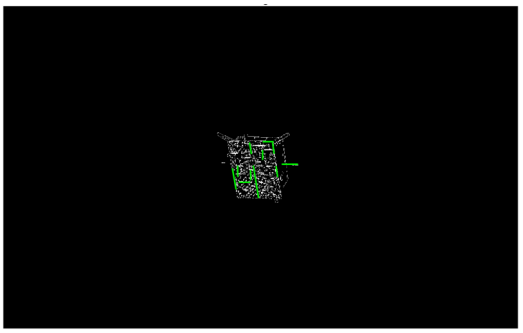


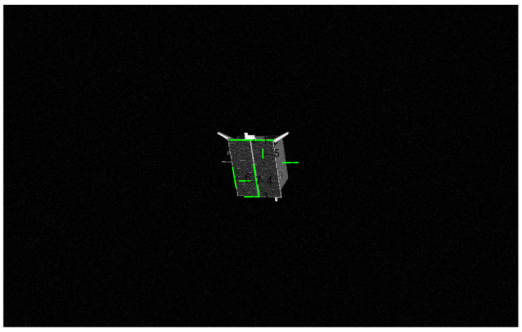
Figure 4.12: Edge detection on figure 4.6e



(a) WGE and Hough result



(b) S&H result

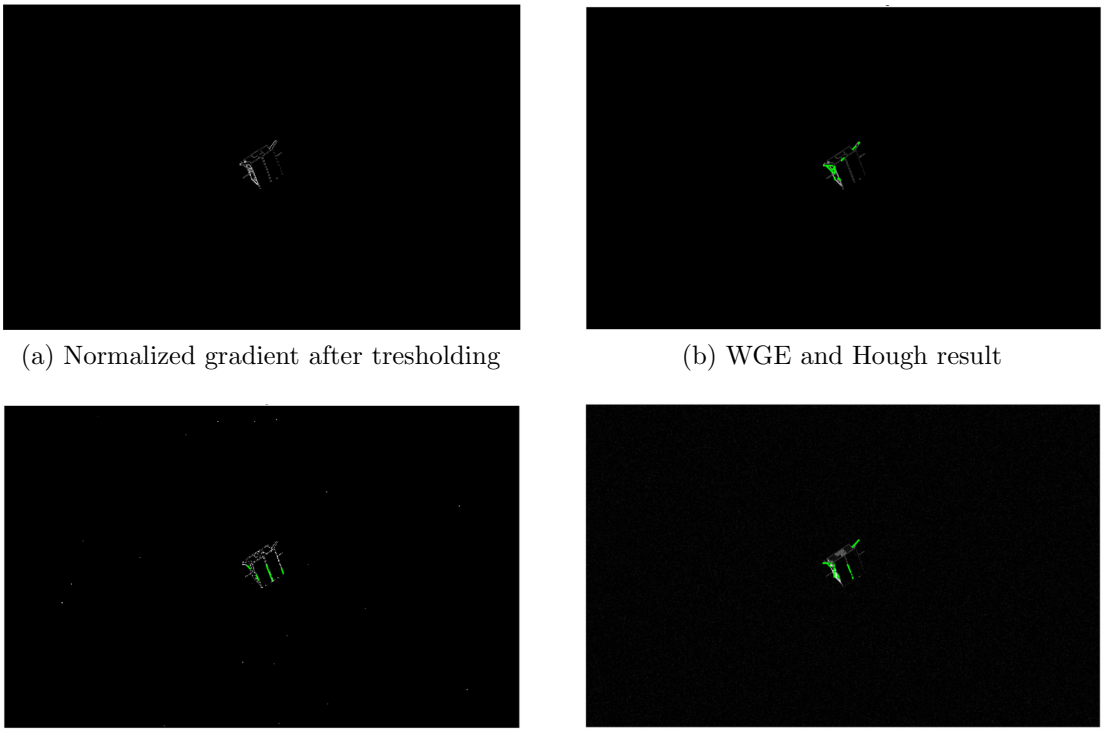


(c) Final result

Figure 4.13: Image where the image processing subsystem contained spurious edges

$$\mathbf{A_{TC}} = \begin{bmatrix} 0.8160 & -0.2675 & 0.5120 \\ -0.5348 & -0.6850 & 0.4943 \\ 0.2183 & -0.6776 & -0.7021 \end{bmatrix},$$

$$\mathbf{t_C} = [-3.6065\ 11.1939\ 11.5997]\ m\,.$$



(a) Normalized gradient after tresholding

(b) WGE and Hough result

(c) S&H result

(d) Final result

Figure 4.14: Image where the image processing subsystem contained not correctly detected edges

That result is likely to be due to the fact that the initial blur done to in order to filter out noise makes edges of the S/C harder to distinguish for the Hough transform if the target is relatively far from the camera, especially for some particular attitudes like the one shown. The result is that the algorithm results will loose accuracy as the relative distance grows.

When a perfect match is give to the pose estimator the mean error are *FARE SIMULAZIONE*

Conclusions

"If you didn't get angry and mad and frustrated, that means you don't care about the end result, and are doing something wrong."

Greg Kroah-Hartman

This project was successful in developing a tool which allows the end user to produce hundreds of images of a target S/C given its STL model, using the open source ray tracer POV-Ray in conjunction with MATLAB. The full uncontrolled dynamics of the target S/C has been simulated as well, so, if the inertial properties of the target are known, is possible to closely simulate the dynamical behavior of the target itself and from that, generate the images. Optionally is possible to also insert the Earth in the background, which enhances the realism of the scene. In particular, a great care has been putted into taking in consideration both the optical properties of the Earth surface and of the S/C surface. The generated data-set has been then compared to the freely available SPEED data-set both in terms of a qualitative assessment (by comparing the histograms) and by applying a CV algorithm. The preliminary results obtained from the comparison shows that similar results have been obtained. Still, there is room for improvement. Such as enhancing the model of the Earth to make it comparable with what found on the SPEED data-set. For what concerns the model of the S/C instead, more realistic texture could be used, if available. A better modeled surface would result in more accurate images produced and a better study could be made to better model the optical properties of the materials which covers the S/C. Moreover, overall rendering times when the Earth is in background are really long, as Earth rendering is a high demanding application. A possible improvement could be to find a way to cut the texture and Earth using not the full sphere but a subset with a mesh, for example. Or better, to patch the ray tracer source code to allow performing the rendering step on the GPU. Concerning the image analysis instead, different pose solvers could be tried. In this work, a MATLAB builtin function has been used. Different pose solver could be used, such as the eP-n-P [38] to make a comparison in terms of efficiency. Another improvement could be to employ separated Hough transforms to detect different geometric shapes, as suggested also in [60]. In general, the image processing subsystem still has to be improved, as it

is susceptible to producing spurious edges, especially when the target is far distant from the camera or when the solar panel are not in shadow (so, when they are hitted directly from the Sun's light). Imagining to implement the algorithm on an embedded board, the proposed toolbox also allows the set-up of hardware in the loop experiments to evaluate the computational time on real H/W. Moreover, having available a toolbox capable of producing hundreds of images at will, also enables to implement completely different strategies for pose determination, such as the one based on neural networks like for example what proposed in [59] and compare the results with what obtained implementing more traditional techniques such as the ones used in this project.

Bibliography

[1] A. F. Abad, O. Ma, K. Pham, and S. Ulrich. A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences*, 68:1–26, jul 2014.

[2] K. T. Alfriend, S. R. Vadali, P. Gurfil, J. P. How, and L. S. Breger. *Spacecraft Formation Flying*. Elsevier, 2009.

[3] J. Arvo. Fast random rotation matrices. In *Graphics Gems III (IBM Version)*, pages 117–120. San Francisco, 1992.

[4] M. Attia, Y. Slama, and M. A. Kamoun. On performance evaluation of registration algorithms for 3d point clouds. In *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*. IEEE, March 2016.

[5] F. H. Bauer, J. O. Bristow, J. R. Carpenter, J. L. Garrison, K. R. Hartman, T. Lee, A. C. Long, D. Kelbel, V. Lu, J. P. How, F. Busse, P. Axelrad, and M. Moreau. Enabling spacecraft formation flying in any earth orbit through spaceborne gps and enhanced autonomy technologies. *Space Technology*, 20(4):175–185, 2001.

[6] C. Beierle and S. D'Amico. Variable-magnification optical stimulator for training and validation of spaceborne vision-based navigation. *Journal of Spacecraft and Rockets*, 56(4):1060–1072, July 2019.

[7] C. Bonnal, J.M. Ruault, and M.C. Desjean. Active debris removal: Recent progress and current trends. *Acta Astronautica*, 85:51–60, apr 2013.

[8] R. Brochard, J. Lebreton, C. Robin, K. Kanani, G. Jonniaux, A. Masson, N. Despré, and A. Berjaoui. Scientific image rendering for space scenes with the surrender software. *ArXiv*, 2018.

[9] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, jun 1986.

[10] M. Casti, A. Bemporad, S. Fineschi, G. Capobianco, D. Loreggia, V. Noce, F. Landini, C. Thizy, D. Galano, and R. Rougeot. PROBA-3 formation-flying metrology: algorithms for the shadow position sensor system. In Nikos Karafolas, Zoran Sodnik, and Bruno Cugny, editors, *International Conference on Space Optics — ICSO 2018*. SPIE, July 2019.

[11] X. Clerc and I. Retat. Astrium vision on space debris removal. In *Proceeding of the 63rd International Astronautical Congress (IAC 2012), Napoli, Italy*, volume 15, 2012.

[12] Wikimedia Commons. The hough transform. `https://upload.wikimedia.org/wikipedia/commons/e/e6/R_theta_line.GIF/`. Accessed: 13-02-2021.

[13] Wikimedia Commons. Ray tracing. `https://upload.wikimedia.org/wikipedia/commons/8/83/Ray_trace_diagram.svg/`. Accessed: 13-02-2021.

[14] A.I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):615–628, July 2006.

[15] Wikipedia Contributors. Nimbostratus cloud. `https://en.wikipedia.org/wiki/Nimbostratus_cloud/`. Accessed: 13-02-2021.

[16] S. D'Amico, , M. Benn, and J. L. Jørgensen. Pose estimation of an uncooperative spacecraft from actual space imagery. *International Journal of Space Science and Engineering*, 2(2), 2014.

[17] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. SoftPOSIT: Simultaneous pose and correspondence determination. *International Journal of Computer Vision*, 59(3):259–284, September 2004.

[18] J. Davis. Mathematical modeling of earth's magnetic field. Technical report, Virginia Tech, Blacksburg, VA 24061, 5 2014.

[19] M. D'Errico, editor. *Distributed Space Missions for Earth System Monitoring*. Springer New York, 2013.

[20] M. Dhome, M. Richetin, J.-T. Lapreste, and G. Rives. Determination of the attitude of 3d objects from a single perspective view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, 1989.

[21] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun. Carla: An open urban driving simulator. *ArXiv*, 2017.

[22] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, July 2002.

[23] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, jan 1972.

[24] 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.

[25] P. Fracchiolla. Analysis and varlidation of a vision-based pose initialization algorithm for non-cooperatie spacecrafts. Master's thesis, Università degli Studi di Padova, Padova, 2019.

[26] XS Gao, XR Hou, J. Tang, and HF Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, aug 2003.

[27] S. Gold, C. P. Lui, A. Rangarajanl, S. Pappul, and E. Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. pages 957–964, 1994.

[28] A. A. Grompone. *Vision-Based 3D Motion Estimation for On-Orbit Proximity Satellite Tracking and Navigation.* PhD thesis, Naval Postgraduate School, 2015.

[29] J. Guarneri. Autonomous optical navigation and attitude estimation system tested on artificially generated images. Master's thesis, Politecnico di Milano, via La Masa 34, Milano, Italia, 7 2020.

[30] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[31] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, USA, 2 edition, 2003.

[32] P V.C. Hough. Method and means for recognizing complex patterns. 12 1962.

[33] SM Parkes I. Martin, M. Dunstan. Planet and asteroid natural scene generation utility final report. Technical Report UoD-PANGU-Final, University of Dundee, Nethergate, Dundee DD1 4HN, Regno Unito, apr 2003.

[34] J. Jensen. Hough Transform for Straight Lines. Technical report.

[35] M. Kaufmann. *An Introduction to Ray Tracing.* Elsevier, 1989.

[36] M. Kisantal, S. Sharma, T. Ha Park, D. Izzo, M. Märtens, and S. D'Amico. Satellite pose estimation challenge: Dataset, competition design and results. *CoRR*, 2019.

[37] R.D. Langley. Apollo Experience Report - The Docking System. 1972.

[38] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o(n) solution to the pnp problem. 81(2):155–166, feb 2009.

[39] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, March 1987.

[40] P. Lunghi. *Hazard Detection and Avoidance Systems for Autonomous Planetary Landing*. PhD thesis, Politecnico di Milano, via La Masa 34, Milano, Italia, 1 2018.

[41] F. L. Markley and J. L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. Springer New York, 2014.

[42] P. Martinez-Gonzalez, S. Oprea, A. Garcia-Garcia, A. Jover-Alvarez, S. Orts-Escolano, and J. Garcia-Rodriguez. Unrealrox: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. *Virtual Reality*, 24:271–288, 2019.

[43] F. M. Mirzaei and S. I. Roumeliotis. Globally optimal pose estimation from line correspondences. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011.

[44] B.J. Naasz, R. D. Burns, S. Z. Queen, J. Van Eepoel, J. Hannah, and E. Skelton. The HST SM4 relative navigation sensor system: Overview and preliminary testing results from the flight robotics lab. *The Journal of the Astronautical Sciences*, 57(1-2):457–483, January 2009.

[45] B.J. Naasz, J. Van Eepoel, S.Z. Queen, C.M. Southward II, and J. Hannah. Flight results from the hst sm4 relative navigation sensor system. volume 137, pages 723–744, 2010.

[46] NASA. Blue marble. `https://visibleearth.nasa.gov/`. Accessed: 13-02-2021.

[47] 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, aug 2017.

[48] Gunter's Space Page. The prisma mission. `https://space.skyrocket.de/doc_sdat/prisma.htm`. Accessed: 13-02-2021.

[49] S.M. Parkes, I. Martin, M. Dunstan, and D. Matthews. *Planet Surface Simulation with PANGU.*

[50] V. Pesce, R. Opromolla, S. Sarno, M. Lavagna, and M. Grassi. Autonomous relative navigation around uncooperative spacecraft based on a single camera. *Aerospace Science and Technology*, 84:1070–1080, jan 2019.

[51] A. Petit, E. Marchand, and K. Kanani. Vision-based detection and tracking for space navigation in a rendezvous context. In *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS*, 2012.

[52] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, sep 2004.

[53] Pedro F. Proença and Y. Gao. Deep learning for spacecraft pose estimation from photorealistic rendering. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6007–6013, 2020.

[54] C. Reinbacher, M. Ruther, and H. Bischof. Pose estimation of known objects by efficient silhouette matching. In *2010 20th International Conference on Pattern Recognition*. IEEE, August 2010.

[55] A. Rivolta. *Guidance Navigation Control and Robotics for On Orbit Servicing.* PhD thesis, Politecnico di Milano, via La Masa 34, Milano, Italia, 1 2019.

[56] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *FSR*, 2017.

[57] S. Sharma. *Pose Estimation of Uncooperative Spacecraft Using Monocular Vision and Deep Learning.* PhD thesis, Stanford University, 450 Serra Mall, Stanford, CA 94305, USA, 9 2019.

[58] S. Sharma and S. D'Amico. Comparative assessment of techniques for initial pose estimation using monocular vision. *Acta Astronautica*, 123:435–445, jun 2016.

[59] S. Sharma and S. D'Amico. Pose estimation for non-cooperative rendezvous using neural networks. *CoRR*, 2019.

[60] S. Sharma, J. Ventura, and S. D'Amico. Robust model-based monocular pose initialization for noncooperative spacecraft rendezvous. *Journal of Spacecraft and Rockets*, 55(6):1414–1429, nov 2018.

[61] JE Solem. *Programming computer vision with Python.* O'Reilly, 2012.

[62] A. Tatsch, N. Fitz-Coy, and S. Gladun. *On-Orbit Servicing: A Brief Survey*, pages 21–23, 2006.

[63] P.H.S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, apr 2000.

[64] J. Ventura. *Autonomous Proximity Operations for Noncooperative Space Target*. PhD thesis, Technical University of Munich, Arcisstrabe 21, Munich, Bavaria, 80333, Germany, 12 2016.

[65] R. Volpe, G. B. Palmerini, and C. Circi. Preliminary analysis of visual navigation performance in close formation flying. In *2017 IEEE Aerospace Conference*. IEEE, mar 2017.

[66] C. Xu, L. Zhang, L. Cheng, and Koch R. Pose estimation from line correspondences: A complete analysis and a series of solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1209–1222, June 2017.

[67] D. Zimpfer, P. Kachmar, and S. Tuohy. Autonomous rendezvous, capture and in-space assembly:past, present and future. In *1st Space Exploration Conference: Continuing the Voyage of Discovery*. American Institute of Aeronautics and Astronautics, jan 2005.

Appendix A

First appendix: Attitude and Orbit Simulator

In this appendix the kinematic and dynamic representation used for the simulation of the attitude of the target spacefrat will briefly be descripted.

A.1 Keplerian Motion

The dynamics of the CG of a rigid body is given by Newton equation of motion stating that the variation in time of the linear mentum is equal to the external forces applied to the body, with respect to an inertial reference frame. Thus :

$$\frac{d}{dt}(m\mathbf{v}) = \sum_{i=1}^{n} \mathbf{f_i} \tag{A.1}$$

where m is the mass of the body, $\mathbf{v}$ its velocity and $\mathbf{f_i}$ the forces applied to the system. Then, for a rigid body orbiting around a single massive body, the main force to take into account is the gravitational pull that can be modelled accordingly to Newton law of gravitation as follows:

$$m\ddot{\mathbf{r}} = -\frac{\mu m}{\|\mathbf{r}\|^3}\mathbf{r} + \mathbf{f} \tag{A.2}$$

where μ is the gravitational constant of the main attractor, $\mathbf{r}$ the position vector of the orbiting body computed from the main attractor CG and $\mathbf{f}$ the sum of other forces applied to the system. The underlying assumption is to consider the main attractor to be still or moving in linear constant motion, which is never the case. Such assumptions holds well enough for many problem of interest. Considering null or negligible other forces it follows that the system motion is central, thus the angular momentum is conserved.

Such quantity, for this system, is defined as follows

$$\mathbf{h} = \mathbf{r} \wedge \mathbf{r} \tag{A.3}$$

where $\mathbf{v}$ is the velocity, derivative of the position vector $\mathbf{r}$ expressed in the said reference frame.
Crossing A.2 with the angular momentum and considering that the gravitational field is conservative, it is possible to determine the position of the orbiting body in time through six parameters that represent the analytical solution of the restricted two body problem. For this research the influence of other massive bodies is not taken into account as for many of the applications here considered can be analysed considering satellites to be small bodies close to the planet.
The most used set of parameters are often referred to as Keplerian parameters, however different parametrization are possible. Of these six constant two represent the shape of the orbit, which must be a conic according to Kepler's studies and Newton's formulation, three identify the orientation of the orbit in a 3-D space and the last one links the position along the orbital with time. The shape is identified by the semi-major axis a of the conic and the eccentricity e, restricted to be between zero and one for closed orbits, being zero for circular orbits. In the reference frame with z axis aligned with angular momentum and x axis aligned with the minimum orbital distance (eccentricity vector) the position vector can be written as follows

$$\mathbf{r_{orb}} = \frac{\|h\|^2/\mu}{1 + ecos\theta} \begin{Bmatrix} cos\theta \\ sin\theta \\ 0 \end{Bmatrix} = \frac{a(1-e^2)}{1 + ecos\theta} \begin{Bmatrix} cos\theta \\ sin\theta \\ 0 \end{Bmatrix} \tag{A.4}$$

Then the position vector $\mathbf{r_{orb}}$ in this frame can be translated back into the inertial $\mathbf{r}$ by using three sequential rotations in the order z - x - z according to the values of the other three parameters: argument of pericenter ω , inclination i and right ascension of the ascending node Ω.

A.2 Euler Equations

The dynamics of a rigid body which is tumbling into space can be modeled by mean of the well known Euler equations, having in mind that such equation is expressed in a non-inertial reference frame attached to the body frame :

$$\dot{\omega_{\mathbf{T}}} = (\mathbf{I_T})^{-1} \left[\mathbf{L_T} - \omega_{\mathbf{T}} \wedge (\mathbf{I_T}\omega_{\mathbf{T}})\right] \tag{A.5}$$

where ω_T is the angular velocity vector of the spacecraft in body frame, I_T is the inertia tensor of the spacecraft in body frame and L_T are the external torques acting on the spacecraft due to external disturbances and control torques.

A.3 Direct Cosine Matrix

The direct cosine matrix, or attitude matrix, gives the transformation of a vector from a reference frame N to another reference frame O :

$$\mathbf{r_O} = \mathbf{A_{ON}}\mathbf{r_N} \tag{A.6}$$

So, if we consider the spacecraft body frame and the inertial frame, then we can write the relation between the two frames as :

$$\mathbf{r_T} = \mathbf{A_{TN}}\mathbf{r_N} \tag{A.7}$$

As it is demonstrated in [41], we can express the time dependence of the attitude matrix by writing the rotation from the body frame to the inertial frame as :

$$\dot{\mathbf{A}}_{\mathbf{TN}} = -[\omega_{TN}\wedge]\mathbf{A_{TN}} \tag{A.8}$$

where $[\omega_{TN}\wedge]$ is the skew-symmetric cross product matrix containing the components of the angular velocity vector :

$$[\omega_{\mathbf{TN}}\wedge] = \begin{bmatrix} 0 & -\omega_{TN_3} & \omega_{TN_2} \\ \omega_{TN_3} & 0 & -\omega_{TN_1} \\ -\omega_{TN_2} & \omega_{TN_1} & 0 \end{bmatrix}$$

Thus, by integrating this relation, we can get full attitude at every iteration.

A.4 Environmental Disturbances

The external disturbances acting on spacecraft placed in a LEO orbit are basically four and are due to :

- Gravity Gradient
- Magnetic Field
- Solar Radiation Pressure
- Aerodynamic Drag

Details about how those torques have been mathematically modeled will be given in the following sections.

Gravity Gradient

Any non-symmetrical rigid body in a gravity field is subject to a gravity-gradient torque. If we consider a rigid spacecraft, the torque due to the gravity gradient about the spacecraft's center of mass can be modeled as :

$$\mathbf{L_{gg}} = \frac{3\ mu}{r^3}\mathbf{n} \wedge (\mathbf{I}\ \mathbf{n}) \tag{A.9}$$

where μ is the Eart's gravitational constant, $\mathbf{r}$ is the radius vector from the center of the Earth, thus $r \equiv \|\mathbf{r}\|$, $\mathbf{I}$ is the inertia tensor in body frame and $\mathbf{n}$ is the body frame representation of a nadir-pointing unit vector.
Further details about the model can be found in reference [41].

Earth's Magnetic Field

The torque generated by a magnetic dipole $\mathbf{m}$ in a magnetic field $\mathbf{B}$ is

$$\mathbf{L_{mag}} = \mathbf{m} \wedge \mathbf{B} \tag{A.10}$$

where $\mathbf{B}$ is the magnetic field in the body frame. The most basic source of a magnetic dipole is a current loop. A current of I amperes flowing in a planar loop of area A produces a dipole moment of magnitude m=IA in the direction normal to the plane of the loop and satisfying a right-hand rule. When $\mathbf{m}$ is in Am^2 and the magnetic field is specified in Tesla, Eq. A.10 gives the torque in Nm. If there are N turns of wire in the loop, the dipole moment has magnitude m=NIA (such as the case of a magnetic torquer). To model $\mathbf{B}$ either the full IGRF model or the simple dipole model can be used.
For what concerns this work, the full IGRF model truncated to the 10th order has been used. Further details about the model can be found in reference [18]

Solar Radiation Pressure

Solar radiation pressure acting on the surfaces of the spacecraft causes a disturbance torque that in general, cannot be neglected for orbits higher than 800 km, so it has been taken into account in this work. The SRP torque is zero zero when the spacecraft is in the shadow of the Earth or any other body, of course. To take into account the effect of solar radiation pressure on the spacecraft, the spacecraft's surface can be modeled as a collection of N flat plates of area S_i, outward normal $\mathbf{n_b}$ in the body coordinate frame, specular reflection coefficient ρ_s, diffuse reflection coefficient ρ_d and absorption coefficient ρ_a; those coefficients must sum to unity. For what concerns this work, only ρ_s and ρ_d have been considered, since all the absorbed radiation is emitted as thermal radiation, although not necessarily at the same time or from the same surface as its absorption. For an accurate modeling of thermal radiation we must also known the absolute temerature and the emissivity

of each surface. We can define the spacecraft-to-Sun unit vector in the spacecraft's body frame as :

$$\mathbf{s_t} = \mathbf{A_{TN}} \mathbf{s_i} \tag{A.11}$$

where $\mathbf{A_{TN}}$ is the attitude matrix and $\mathbf{s_i}$ is the spacecraft-to-Sun unit vector in the GCI frame. We can define the angle between the Sun vector and the normal exiting from the normal to the i-th plate as :

$$cos(\theta^i_{SRP}) = \mathbf{n}^i_T \cdot \mathbf{s_b} \tag{A.12}$$

The SRP force on the i-th plate can be expressed as :

$$\mathbf{F}_{SRP} = -P_{Sun} A_i \left[2 \left(\frac{\rho^i_d}{3} + \rho^i_s cos(\theta^i_{SRP}) \right) \mathbf{n}^i_B + (1 - \rho_s) \mathbf{s_t} \right] max(cos(\theta^i_{SRP}), 0) \tag{A.13}$$

where P_{Sun} is the solar radiation pressure. The Solar radiation pressure torque acting on the spacecraft is then given by :

$$\mathbf{L}^i_{SRP} = \sum_{i=1}^{n} \mathbf{r}_i \wedge \mathbf{F}^i_{SRP} \tag{A.14}$$

where $\mathbf{r}_i$ is the vector from the spacecraft center of mass to the centre of pressure of the SRP on the i-th face. In this formulation we are not considering the albedo radiation coming from the Earth and from the Moon. Further details on how the solar radiation pressure, the spacecraft-to-Sun unit vector and the eclipse condition have been modeled can be found in reference [41].

Aerodynamic Drag

Aerodynamic torque is generally computed by modeling the spacecraft as a collection of N flat plates of area A_i and outward normal unit vector $\mathbf{n_B}$ expressed in the spacecraft body-fixed coordinate system. The torque depends on the velocity of the spacecraft relative to the atmosphere, which is not simply the velocity of the spacecraft in the GCI frame, because the atmosphere is not stationary in that frame. The most common assumption is that the atmosphere co-rotates with the Earth. The relative velocity in the GCI frame is then given by :

$$\mathbf{v_{relI}} = \mathbf{v_I} + [\omega_{\oplus} \wedge] \mathbf{r_I} \tag{A.15}$$

where $\mathbf{r_I}$ and $\mathbf{v_I}$ are the position and velocity of the spacecraft expressed in the GCI frame. The Earth's angular velocity vector is $\omega_{\oplus} = \omega_{\oplus}[0\ 0\ 1]'$ with $\omega_{\oplus} =$

0.000 072 921 158 553 rad/s. The velocity in the body frame is the computed as :

$$\mathbf{v_{relT}} = \mathbf{A_{TN}} \begin{bmatrix} \dot{x} + \omega_{\Earth} y \\ \dot{y} - \omega_{\Earth} x \\ \dot{x} \end{bmatrix} \tag{A.16}$$

The inclination of the *i-th* plate with respect to the relative velocity is given by :

$$cos(\theta^i_{aero}) = \frac{\mathbf{n}^i_T \cdot \mathbf{v}_{rel\ T}}{\|\mathbf{v}_{rel}\|} \tag{A.17}$$

The aerodynamic force on i-th plate in the flat plate model is

$$\mathbf{F}^i_{aero} = -\frac{1}{2}\rho C_D \|\mathbf{v}_{rel}\| \mathbf{v}_{relT}\ S_i\ max(cos(\theta^i_{aero}), 0) \tag{A.18}$$

where ρ is the atmospheric density, and C_D is the drag coefficient. ρ can be modeled by mean of the well known exponential decaying model for the atmospheric density :

$$\rho = \rho_0 e^{(-(h-h_0)/H)} \tag{A.19}$$

where ρ_0 and h_0 are reference density and height, respectively, h is the height above the ellipsoid and H is the scale height, which is the fractional change in density with eight. The value of ρ_0, h_0 and H changes with h. The values used to perform the simulation are the one given in [41]. The actual torque due to the aerodynamic drag can be computed as :

$$\mathbf{L}^i_{aero} = \sum_{i=1}^{n} \mathbf{r}_i \wedge \mathbf{F}^i_{aero} \tag{A.20}$$

where n is the number of faces and $\mathbf{r}_i$ is the vector from the spacecraft center of mass to the center of pressure on the i th face.

Appendix B

Second appendix: Common CV Models and Problems

In this appendix the reader will be presented with a brief mathematics discussion regarding some common CV models and problems, which could be useful to better understand what is discussed in chapter 3.

B.1 Pinhole Camera Model

The pinhole camera model is camera model which is widely used in Computer-Vision (CV). Despite being a relatively simple model, it is still accurate enough for the vast majority of applications. The pinhole camera model is used to describe the mathematical relationship which exists between the coordinates of a point in the 3-D world and its projection onto the image plane. In the pinhole camera model, the light passes trough a single point, the camera center, before being projected onto the image plane, and no lenses are used to focus light, so geometrical distortions or blurring are not modeled in the pinhole camera model.

By referring to B.1, a 3-D point $\mathbf{X}$ is projected to an image point $\mathbf{x}$ (both expressed in homogeneous coordinates) as :

$$\lambda \mathbf{x} = P\mathbf{X}\,,$$

where P is a 3-by-4 matrix called camera matrix and λ is a scalar number representing the inverse depth of the 3-D point. As a result, the 3-D point $\mathbf{X}$ is characterized by four elements, $X = [X, Y, Z, W]$ in homogeneous coordinates. Generally, the camera matrix P can be decomposed as :

$$P = K[R \mid \mathbf{t}]\,,$$

where R is the rotation matrix which describes the orientation of the camera, $\mathbf{t}$ is a 3-D translation vector which describes the position of the camera center

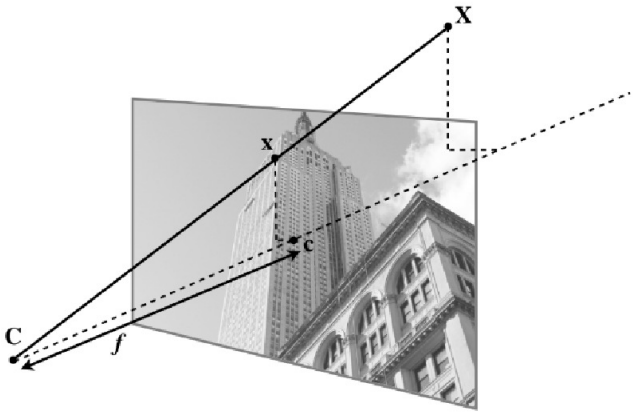


Figure B.1: The pin-hole camera model [61]

and K is the intrinsic camera calibration matrix. The camera calibration matrix basically describes the projection properties of the camera and can be written as :

$$K = \begin{bmatrix} \alpha f & s & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix},$$

where α is the aspect ratio of the sensor's pixels, f is the focal length, s is the skew and c_x, c_y are the coordinates of the optical center (or also called the principal point) of the image. Usually the principal point of the image can be approximated with half the height and half the width of the image. The skew is used only if the pixel array in the sensor is skewed. In most cases is safe to assume $s = 0$. By defining :

$$f_x = \alpha f_y$$

and by neglecting s we can write the calibration matrix in the most common form :

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix},$$

If we make the assumption of square pixel, then $\alpha = 1$ and $f_x = f_y = f$, and so we can write the camera calibration matrix as :

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

More information about camera model and camera calibration matrix can be found in [31] and [52].

B.2 Image Derivatives

For a grayscale image, the image intensity changes over the image itself can be described by using the x and y derivatives, I_x and I_y of the graylevel image I. Once defined the image derivatives, it is possible to define another quantity, the image gradient, which is the vector:

$$|\nabla I| = \sqrt{I_x^2 + I_y^2}\,,$$

which describes how strong the image intensity change. Image derivatives can be computed using a discrete approximations, which are implemented as convolutions:

$$I_x = I * D_x \quad , \quad I_y = I * D_y\,,$$

where D_x, D_y are the kernel of the derivative and $*$ is the 2-D convolutional operation. Two common choices for D_x, D_y are either the so called Prewitt filters :

$$D_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad , \quad D_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}\,,$$

or the so called Sobel filters:

$$D_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad , \quad D_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}\,,$$

B.3 The Hough Transform

The Hough transform is a method often used in CV for finding shapes in images. It works by using a voting procedure in the parameters space of the shapes. The most common use of the Hough transform is to find lines in images. Consider the straight line equation, in its polar form,

$$\rho = x\cos\theta + y\sin\theta\,,$$

where ρ obviously is the distance from the origin of the reference system to the closest point on the straight line, while θ is the angle between the x axis and the line connecting the origin with that closest point. The linear Hough transform algorithm estimates the two parameters that define the straight line. The transform space has two dimensions, and any point in the transform space is used as an accumulator (a bi-dimensional vector) to detect a line described by the aforementioned polar equation of the straight line. Every point in the detected edges in the image contributes to the accumulators. Generally, the dimension of

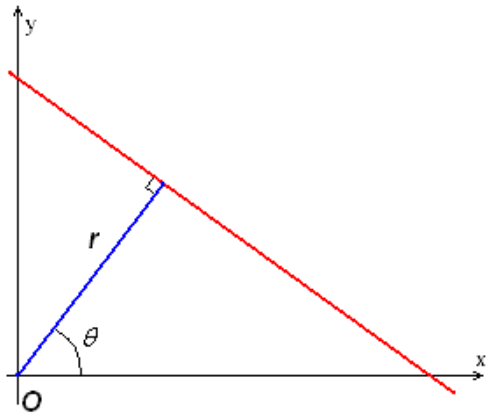


Figure B.2: Polar representation of a straight line [12]

the accumulator is equal to the number of unknown parameters, which in this case are two, ρ and θ. For each pixel and is neighborhood, the algorithm which computes the Hough transform first try to detect if the specific pixel which is being analyzed lie on a line. If so, computes ρ and θ of that line, and then look for the accumulator's bin that the parameters fall into, and increment the value of that bin. By searching the boxes with highest values, typically by looking at local maxima in the accumulator space, it is possible to identify the the most likely lines. The final result of the Hough transform will be a 2-D matrix, were one dimension will be represented by θ and the other one by ρ. Each element of that matrix will have a value equal to the sum of the pixels that are positioned on the line represented by the parameters ρ, θ. So the element with the highest value indicates the straight line that is most represented in the input image [34]. More information about the how the Hough transform works can be retrieved in [23] and [32].

B.4 Perspective-n-Point Problem

The Perspective-n-Point (P-n-P) point is the problem of determining the position and the orientation of a calibrated camera given its intrinsic parameters and a set of correspondences between a given set of 3-D points and their respective 2-D projections in the image [38]. The perspective projection for a standard pinhole camera that has been introduced in section B.1 leads to the following equation (in homogeneous coordinates) for the model:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix},$$

were r_{ij} and t_i are the components of the rotation matrix and the translation vector which are being calculated, respectively. Several methods exists for solving the P-n-P problem, the two most common of which are the P3P method and the eP-n-P method. More information about their implementations can be found on [26] [63] and [38].