UNIVERSIDAD AUTONÓMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Spacecraft Pose Estimation For Non-Cooperative Rendezvous Based On Feature Codebook

Máster Universitario en Investigación e Innovación en TIC (I^2-TIC)

Autor: Bravo Pérez-Villar, Juan Ignacio

Tutores:
Escudero Viñolo, Marcos (UAM)
Kerr, Murray (Deimos Space S.L.U.)

Ponente:
Martínez Sánchez, Jose María

Septiembre, 2017

UNIVERSIDAD AUTÓNOMA DE MADRID

Spacecraft Pose Estimation For Non-Cooperative Rendezvous Based On Feature Codebook

by

Juan Ignacio Bravo Pérez-Villar

Advisors:
Marcos Escudero Viñolo (UAM)
Murray Kerr (Deimos Space)

Escuela Politécnica Superior

September 2017

"Most people stare up into space with wonder. Yet, we have this almost-alien world on our own planet just teeming with life."

Jeff Orlowski

UNIVERSIDAD AUTÓNOMA DE MADRID

Abstract

The goal of this master's thesis is the implementation and analysis of an algorithm able to detect the pose of a non-cooperative spacecraft in autonomous rendezvous scenarios. The algorithm is based on a codebook approach, where features reduced in dimensionality with a two-dimensional principal component algorithm are associated to attitudes. The training set of attitudes is generated using a 3D-CAD model of the non-cooperative spacecraft, with the rotations in the 3D space defined by means of random uniform quaternions.

The document begins with a review of the different methods for spacecraft pose estimation under non-cooperative rendezvous scenarios, the Principal Component Analysis with some of its two-dimensional variations, and a study of the methods for sampling all the possible orientations of a 3D-object. Next, the proposed architecture of the system is described and tested, over a set of tests designed to simulate the different possible environments in a real case of application. The document is concluded with the discussed results, conclusions and guidelines for future work.

The proposed algorithm allows to estimate accurately the pose under different illumination conditions and distances to the spacecraft, with a moderate computational load and without previous knowledge of the spacecraft's pose.

Keywords: codebook, 3D-Model, histogram of oriented gradients, monocular camera, non-cooperative, pose, principal component analysis, quaternions, regularization, rendezvous, spacecraft, sparsity.

UNIVERSIDAD AUTÓNOMA DE MADRID

Resumen

El objetivo de este Trabajo de Fin de Máster es la implementación y análisis de un algoritmo capaz de detectar la pose de un satélite no cooperativo, para operaciones de proximidad entre dos naves espaciales autónomas. El algoritmo se basa en un *codebook* formado por descriptores, reducidos en dimensionalidad con una técnica de componentes principales bidimensional, asociados a orientaciones. El set de imágenes de entrenamiento, del cual se extraen los descriptores, es generado rotando mediante quaterniones un modelo 3D de la nave no cooperativa y renderizando las imágenes con un entorno 3D.

El documento comienza con una revisión de los diferentes métodos existentes para detectar la pose de una nave espacial no cooperativa enfocados a escenarios de proximidad. Continúa con la introducción de algunos de los algoritmos de análisis de componentes principales y el estudio de algunos de los métodos para extraer representaciones bi-dimensionales de objetos 3D. A continuación, se presenta y explica la arquitectura del sistema para proseguir con los experimentos y resultados obtenidos. El documento finaliza con las conclusiones y las líneas propuestas de trabajo futuro.

El algoritmo propuesto consigue estimar la pose de forma precisa bajo diferentes condiciones de iluminación y distancias a la nave no cooperativa, sin necesidad de conocer la pose en instantes anteriores y con una carga computacional moderada.

Palabras clave: análisis de componentes principales, cámara monocular, codebook, histograma de gradientes orientados, modelo 3D, nave espacial, no cooperativo, pose, proximidad, quaterniones, regularización, selección de variables.

Acknowledgements

First of all, I would like to thank Virginia Fernández for proposing the topic of this master's thesis and introducing me into this field. Thanks to Murray Kerr and Paulo Rosa for having me in your team, and give me the liberty of creating the result of this master's thesis. Thanks to Aniello Fiengo for the support and advices, and to my fellow interns Pablo Moral and Enrique Santiago for making the days better in the office.

Special thanks to my principal advisor Marcos Escudero Viñolo. Thanks for accepting this unusual project, your constant interest, and space conversations at the end of each session. Thank you for the advices, ideas and effort that helped to improve this work and taught me during the path.

Thanks to my family for the unconditional support and the means that made all this possible. Specially to my mother, your efforts were essential to bring me here.

I would also like to thank Grant Borodin for allowing me to use his Envisat 3D-Model.

Lastly, thanks to you reader, I hope you may find this document useful.

Contents

List of Figures

List of Tables

Symbols

$\mathbf{y}$	Feature column vector
$\mathbf{Y}$	Data matrix formed by concatenation of column feature vectors
$\mathbf{x}_i^j$	j-th row from data matrix $\mathbf{X}_i$
$\mathbf{X}$	Data matrix conformed by concatenation of data matrices
λ	Eigenvalue
$\mathbf{\Lambda}$	Eigenvalue matrix
$\mathbf{\Psi}$	Eigenvector matrix
$\mathbf{Q}$	Covariance matrix
$\mathbf{S}$	Inner-scatter matrix
$\mathbf{U}$	U unitary matrix from SVD decomposition
$\mathbf{V}$	V unitary matrix from SVD decomposition
$\mathbf{\Sigma}$	Singular value matrix from SVD decomposition
$\mathbf{P}$	Projection matrix
$\mathbf{G}$	Projected feature matrix
$\mathbf{v}$	Principal component vector
$\mathbf{u}$	Projected principal component vector
L	PCA objective function
J	2DPCA objective function
Z	2DPCAL_1 objective function
R	Sparse 2DPCAL_1 objective function
s	Absolute value auxiliary function
η	Regularisation parameter
γ	Regularisation parameter
k	Number of principal components
Q	Quaternion
Γ_x	Rotation matrix
r	Relative height

$\mathbf{I}$	Pixel intensity descriptor
$\mathbf{H}$	Histogram of Oriented Gradients Descriptor
$\mathbf{D}^P$	Projected final descriptor
N	Number of images in test dataset
μ	Dot product between two quaternions
MQE	Mean Quaternion Error
$dist_{\%}$	Percentage of distances over a threshold

Chapter 1

Introduction

1.1 Motivation

Orbital debris is becoming a major problem for space system operations in near Earth orbits. According to the European Space Agency (ESA) estimations, currently there are more than 29,000 objects bigger than 10 cm orbiting the Earth. Among all the debris objects, the decommissioned satellites in near Earth orbits are of high interest, as a collision with them could lead to the generation of a large debris cloud that may grow indefinitely. The delay in earth-space communication disregards the use of remote control operations for the disposal of debris objects. To avoid the risk involved in the use of manned spacecraft, the capture and controlled burn of large debris objects must be done autonomously by an unmanned spacecraft.

In this field, the use of computer vision techniques may provide help to estimate the relative position of the autonomous spacecraft and the debris object, or even provide assistance in cases when communication (e.g. via GPS) with reference satellites are lost. It is important to remark that, due to the nature of the spacecraft processors and power limitations, the computational complexity of the potential computer vision algorithms is constrained.

1.2 Context

This section introduces the reader into the framework of this master's thesis. First, the term autonomous rendezvous in space is defined in Section 1.2.1. Next, the particular case of study is analysed in Section 1.2.2. The chapter is concluded in Section 1.2.3 with the challenges and restrictions that the algorithm must overcome.

1.2.1 Autonomous Rendezvous in Space

The term autonomous rendezvous in space refers to an orbital manoeuvre, where an unmanned chasing spacecraft approximates to a target travelling in space. The target, usually an artificial satellite or space station, remains passive, whereas the chaser executes relative navigation operations [4].

The target can be considered cooperative or non-cooperative. Cooperative targets either communicate its position to the chaser or provide visual markers to guide the proximity operations. Non-cooperative targets do not provide any information, i.e. the chaser depends on autonomous relative navigation technologies.

The chaser-target rendezvous operation in a space scenario can be split into five main stages (see Figure 1.1)

- **i** Separate orbits: In this stage, both the chaser and target are in entirely different orbits. The chaser spacecraft must have approximate knowledge of the orbit and position of the target.
- **ii** Drift orbit i: The chaser approaches the target orbit and drifts towards it. Typically, it finishes below and behind the target, on a close but lower orbit. This allows to avoid collisions, and confers a higher velocity to the chaser.
- **iii** Drift orbit ii: In this stage, the chasing spacecraft has visibility of the target. This range, typically between 3,000 and 10,000 km for Low Earth Orbits (LEOs) [4] allows communication between spacecraft, or relative autonomous navigation in non-cooperative scenarios.
- **iv** Proximity operations A: The distance range in this stage extents from 1km to 100m. The chaser uses relative navigation systems to approach the target through small thruster firings.
- **v** Proximity operations B: This stage, in which the spacecraft are from 10 to 100 m apart, is the last for inspection missions and the previous one for docking. In inspection missions the chaser is positioned within a small distance to the target, matching the attitude[1] or circumnavigating it. In docking missions, the chaser must match the attitude and position of the target, drifting towards the docking platform.

[1]The term attitude refers to the rotational orientation of a spacecraft in space with respect to a known frame.

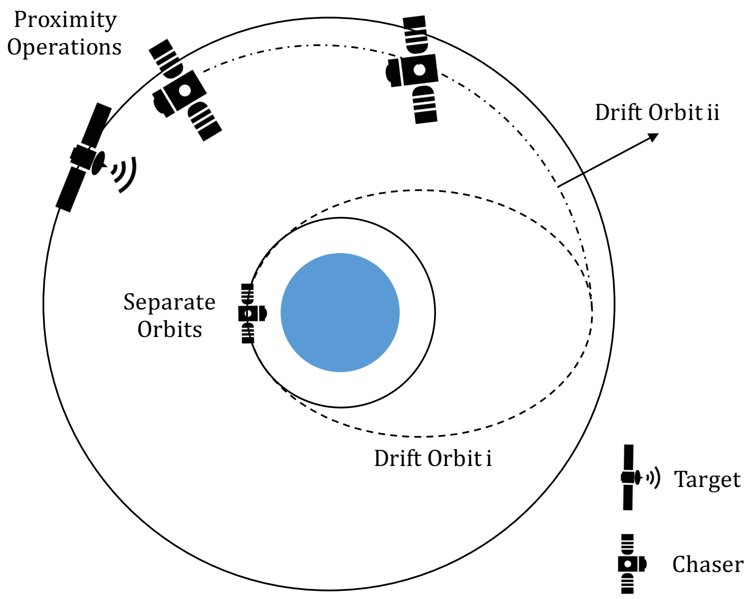

FIGURE 1.1: Non-Cooperative Rendezvous Stages. The blue circle represents the Earth

1.2.2 Case of Study: Envisat

Relative navigation in proximity operations is increasing in significance as a result of its wide applicability in modern space missions. Potential applications include: orbital debris removal, re-supplying, structure formation, inspection, and docking.

The objective of this master's thesis is to develop a pose (attitude + location) estimation algorithm suitable for non-cooperative rendezvous missions. To motivate the work and provide a study scenario, this master's thesis considers the e.Deorbit mission, to be launched by the ESA in 2023 [5]. This mission, which is part of the ESA Clean Space Initiative, will remove from orbit a decommissioned ESA-owned spacecraft, aiming to avoid the generation of more space debris and to prevent its fall into the Earth.

Envisat (see Figure 1.2) is the target chosen to be removed by the ESA due to its high environmental risk. It has a large cross-section of $\pm 64\text{m}^2$, a large mass of ± 8 tonnes and it is exposed to the LEO high debris flux [6]. It is also a good study case, as significant material is publicly available for this spacecraft and the e.Deorbit mission.

A collision with the Envisat could potentially generate a large debris cloud which can collide in turn with other satellites, generating a cascade effect known as Kessler syndrome. In this scenario, the population of space debris will increase, even without new launches. This poses a problem for the already existing satellites in orbit, manned stations such as the International

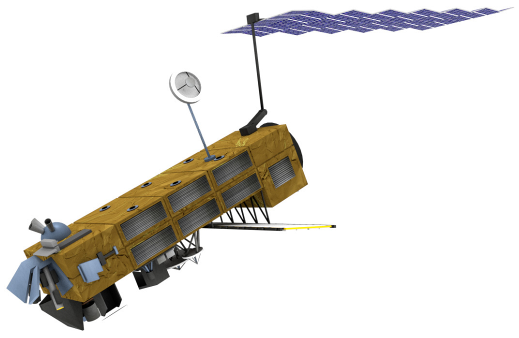

FIGURE 1.2: Envisat image from 3D model (Grant Borodin)

Space Station (ISS) and new possible launches, considering that the space debris can reach speeds of 25.000 - 29.000 Km/h.

Besides, due its large mass of 8 tonnes, the Envisat may survive the entry into the atmosphere, with the risk of falling onto a populated area [7]. Accordingly, several strategies are under study to capture and make a controlled de-orbit of the Envisat.

The analysis of the challenges and technologies of the mission is out of the scope of this master's thesis, if the reader is interested please refer to [6] and [8] for a detailed analysis of the current mission options.

1.2.3 Challenges and Requirements

The use of image processing techniques to aid in the non-cooperative rendezvous is under consideration. According to ESA [9], the following challenges need to be handled:

- Uncooperative target: The Envisat will act as an uncooperative target which won't provide information about its attitude. In addition, the Envisat does not have visual markers for assistance.
- Fast illumination changes: Envisat is placed in LEO, with an orbital period of 100.16 minutes. This, coupled with the complex shapes and highly reflective materials of the

Envisat, can lead to heterogeneous illumination conditions during the rendezvous. Despite its orbit can be propagated to the year of the capture mission to estimate the position of the Sun with respect to the object, the chaser must be designed to cope with all lightning conditions, as its attitude dynamics of Envisat are unknown [6].

- High symmetry: Can induce ambiguities if rotation invariance is considered for the extracted features, as two different poses can lead to the same visual descriptors.
- Need of real time processing and fast algorithms: The limited amount of computational resources in the chaser spacecraft –due to power requirements and hardware design– entails the need of efficient solutions.

Consequently, the developed algorithm must not expect information provided by the target, be robust to illumination conditions, be capable of solve ambiguities due to the symmetry of the Envisat, and present a low computational load.

1.3 Structure

The structure of the rest of the document is as follows:

- Chapter 2 presents the existing methods for spacecraft pose estimation in non-cooperative rendezvous scenarios. Furthermore, it includes a review of Principal Component Analysis, and some of the different methods for obtaining two-dimensional representations of a three-dimensional object.
- Chapter 3 describes and explains the architecture of the proposed system for the task of spacecraft's pose estimation.
- Chapter 4 introduces the data, metrics, and experiments used to evaluate the goodness of the algorithm, including a detailed discussion of the results.
- Chapter 5 presents a summary and conclusions of the developed work, including outlines for future research.

Chapter 2

Literature Review

This chapter is divided into three main sections. Section 2.1 presents a review of the relative pose estimation algorithms in non-cooperative rendezvous scenarios. Section 2.2 describes the theoretical background of the Principal Component Analysis (PCA), including: two-dimensional PCA, two-dimensional PCA with L_1 norm, and the solution used in this master's thesis, two-dimensional PCA with L_1 norm and sparse regularization. Finally, Section 2.3 reviews some of the existing methods for the task of sampling the orientations of an object in the 3D space.

2.1 Spacecraft Pose Estimation

Spacecraft pose estimation is defined as the process of estimating the rigid transform that aligns a model frame with a sensor, body, or world reference frame [10]. Algorithms for spacecraft pose estimation are divided into the literature in two broad classes: those designed to exploit previous knowledge of the target spacecraft derived from a previously available 3D-model (3D-model based), and those that do not rely on previous 3D-models (Non-Model based).

The Non-Model vein is intended to provide a broader applicability, as the algorithms are not designed for a specific target. They can be further divided into two subclasses:

- Shape tracking: This set of algorithms rely on the assumption that the spacecraft is provided with markers made up of recognizable shapes (e.g. rectangles), and aim to estimate its pose by finding the camera transformation between the observed and the real world shape.

- Feature point tracking: These algorithms are usually designed to detect and track a set of feature points. The estimation of the dynamics of the spacecraft is obtained from measurements between consecutive input frames.

The 3D-Model vein is constrained by the requirement of a 3D-model of the target spacecraft. A sub-classification of these methods can be done based on the underlying technique:

- Matching of 3D points: These algorithms use depth sensors to extract 3D points from the surface of the spacecraft, and align the 3D points with the 3D-model.
- Projection of detected features: The pose is estimated by finding the transformation between the observed features extracted from the data captured by the input sensor, and the features extracted from the 3D-model using Perspective-n-Points solvers.
- Codebook approaches: The input image is compared with a codebook which associates image representations to poses. The pose is retrieved from the entry of the codebook that results in the match of minimum distance to the input image.

2.1.1 Non-Model Based

These approaches may use *a priori* information of the spacecraft, but not extracted from a 3D model representing a specific design. In the absence of information about the spacecraft, the usual process is to detect and track a set of feature points to estimate the dynamics of the spacecraft [11] [12]. If some information is known, then it is generally used to find the transformation between the camera image projection and the real target [1][13].

Among approaches based on shape tracking, in [13] the authors propose a rectangle detection based algorithm for very close range operations (1-5m). To capture the whole spacecraft from such a close distance, a camera with a wide or ultra-wide angle lens is required, hence obtaining a radial-distorted image. To overcome this limitation, they propose the use of two coordinated parallel cameras. The pose is estimated by first resolving the coordinates of the detected rectangle vertices in each camera frame. Later, the information is fused and aligned on a common reference. Finally, the rotation matrix and translation vector of the object is used to define the relative pose.

In [1], the authors propose a similar approach but relying on circular features present in the adapter rings of the spacecraft. Making use of a stereo camera they are able to find the transformation relating the adapter ring with their projected images (see Figure 2.1). The

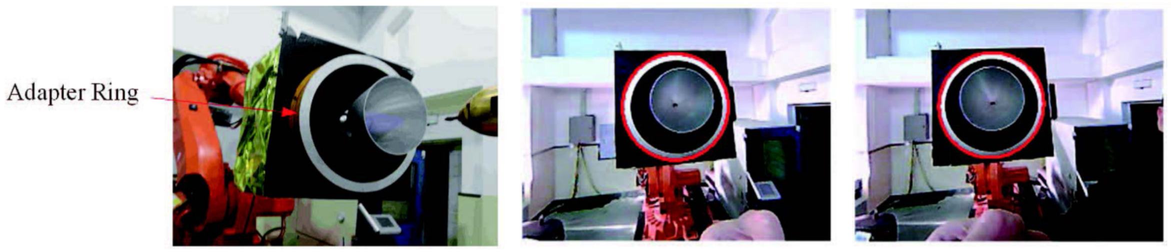

FIGURE 2.1: Adapter ring and ellipse detection. Reproduced from [1].

relative pose is expressed by means of one of the camera's projection matrix, that relates the observed image with the real world.

Shape dependency is removed by feature point tracking techniques. In [11] Segal, Carmi, and Gurfil propose to detect arbitrary feature points on the spacecraft with a stereo camera, and to track them in time by means of Iterative Extended Kalman Filters (IEKF). Another representative example of feature point tracking techniques is in [12], in this study, Lichter and Dubowsky propose a method to track depth feature points whereby on-line build a 3D model. The tracking is driven by Kalman filters for the estimation of the translational and the rotational vectors of the object. Aided with motion information, the new tracked points are used to generate on-line a 3D model of the object.

2.1.2 Model Based

Non-Model based algorithms are useful when there is no model of the target spacecraft, but they are conditioned by the proper operation of feature trackers [3]: whereas the non-model dependency generalizes these solutions, the tracking maybe affected by the lack of model-information (e.g. in its operation in poor illumination conditions and blurred images).

The general idea of 3D-Model based algorithms is to find the transformation between a set of observed features extracted from the input image, and the corresponding features from the stored 3D-Model. Depending on the type of match, a sub-classification can be done.

Several approaches rely on depth information to match 2D and 3D features. One illustrative example can be found in [2]. The authors make use of a 3D model of the target spacecraft and depth measures to estimate the pose of the spacecraft. They propose to combine the Iterative Closest Point (ICP) [14] algorithm with a Kalman filter to predict the next pose and initialize ICP. See Figure 2.2.

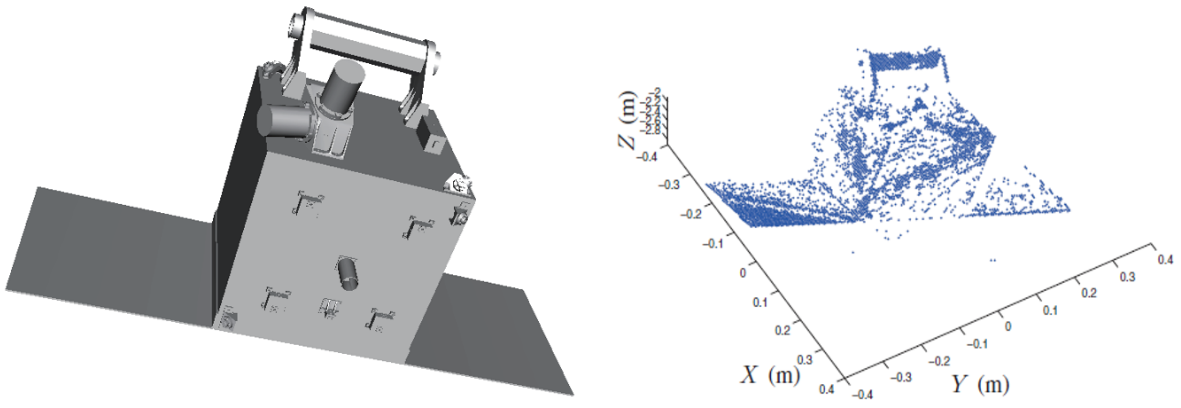

Figure 2.2: 3D-Point cloud acquisition example. Reproduced from [2]

If the sensor used does not allow to capture depth information, e.g. monocular cameras, Perspective-n-Point (PnP) solvers can be used for retrieving the relative pose between target and chaser [15]. Generally, a set of feature points are extracted and described from the input image. The idea is to find the correspondence between the observed feature points extracted from the signal of the input sensor, and the corresponding 3D-points in the stored model. The PnP solver aims to determine the pose of the camera, understanding the observed feature points as 2D image projections from a set of n 3D points, representing some part of the 3D-model.

An example of this approaches can be found in the studies of Petit, Marchand and Kanani [16] [17]. Their pose estimation algorithms are based on finding the transformation on the projection of points extracted from the image edges to a 3D model composed of lines. The use of such inexpensive features allows the authors to achieve real time performance with the use of dedicated hardware.

Methods relying on monocular passive sensors for space imagery can be computationally costly, and sometimes infeasible for real-time applications (e.g SIFT does not usually allow real time performance [15]). In addition, the spatial relationship between features, that implicitly encodes the information of the pose, is not explicitly present. For these reasons Shi, Ulrich and Ruel [3] propose the matching of the image itself rather the matching of features. Their work, to our knowledge, is the only pose estimation algorithm based on a codebook in the literature.

Used as baseline for this master's thesis, [3] is focused on retrieving the pose by comparing image features against a codebook of pre-computed features associated to attitudes. With the use of a spacecraft 3D-model, a subset of images at different attitudes is obtained and described with the images' intensity values. However, as the direct comparison of two images can be computationally expensive, Principal Component Analysis (PCA) [18] are first extracted to reduce the dimensionality of the images. The authors claim that a set of 30 components per image, resulting from projecting the image onto the first 50 principal components is enough to encode the image information. An implemented version of this algorithm is evaluated and compared to the proposed method in Chapter 4.

2.2 Principal Component Analysis

This section serves as a brief review of the PCA algorithm and some of its versions. As the pose estimation algorithm explored in this master's thesis relies on 2DPCA-L_1 with sparse regularization, we first introduce PCA, then continue with 2DPCA and 2DPCA with L_1 norm. Finally, the introduction of sparsity by means of regularization is explored.

2.2.1 PCA

Principal Component Analysis [18], is a statistical procedure used to reduce the data dimensionality, while preserving the variance. The procedure extracts the vectors, known as principal components, representing the directions with most variability of the data. With the constraint that each principal component must be orthogonal to the rest. The data dimensionality is reduced by projecting the data onto the subspace spanned by the principal components.

The estimation of the principal components relies on the covariance matrix of the data. Let $\mathbf{Y}$ be the data matrix, formed by concatenation of M mean-centred data column vectors $\mathbf{y}_j$, and let $\mathbf{Q}$ be the covariance matrix of the data expressed by:

$$\mathbf{Q} = \mathbf{Y}\mathbf{Y}^T \tag{2.1}$$

where,

$$\mathbf{Y} = \begin{bmatrix}\mathbf{y}_1 & \dots & \mathbf{y}_j & \dots & \mathbf{y}_M\end{bmatrix}. \tag{2.2}$$

Let $\mathbf{v}_1$ be the vector representing the direction of the largest variance. Then, the projection of each data point onto this one-dimensional vector is expressed as ${\mathbf{v}_1}^T\mathbf{y}_j$. The variance of the projected data points can be expressed as:

$$\mathbf{v}_1{}^T\mathbf{Q}\mathbf{v}_1. \tag{2.3}$$

So, the vector that maximize Expression 2.3 will represent the direction with the largest variance. Please note this has to be a constrained maximization problem to avoid $\|\mathbf{v}_1\| \to \infty$. An appropriate constraint [19] is to set $\|\mathbf{v}_1\| = 1$, as the interest is in the direction of the data, not its magnitude.

This vector $\mathbf{v}_1$ is the eigenvector associated to the largest eigenvalue. An intuitive proof can be done with the spectral decomposition of $\mathbf{Q}$:

$$\mathbf{Q} = \boldsymbol{\Psi}\boldsymbol{\Lambda}\boldsymbol{\Psi}^T, \tag{2.4}$$

where $\boldsymbol{\Psi}$ represents the matrix corresponding to the orthonormal eigenvectors, and $\boldsymbol{\Lambda}$ a diagonal matrix containing the eigenvalues. Let replace $\mathbf{Q}$ in Expression 2.3 with Equation 2.4.

$$\mathbf{v}_1{}^T\mathbf{Q}\mathbf{v}_1 = \mathbf{v}_1{}^T\boldsymbol{\Psi}\boldsymbol{\Lambda}\boldsymbol{\Psi}^T\mathbf{v}_1, \tag{2.5}$$

and let $\mathbf{u} = \boldsymbol{\Psi}^T\mathbf{v}_1$ be the projected vector $\mathbf{v}_1$ onto the space spanned by the orthonormal eigenvectors $\boldsymbol{\Psi}$. Then, by the orthonormality of $\boldsymbol{\Psi}$ and the constraint $\|\mathbf{v}_1\| = 1$, the projected vector is unitary $\|\mathbf{u}\| = 1$

$$\mathbf{v}_1^T\boldsymbol{\Psi}\boldsymbol{\Lambda}\boldsymbol{\Psi}^T\mathbf{v}_1 = \mathbf{u}^T\boldsymbol{\Lambda}\mathbf{u}. \tag{2.6}$$

The quadratic form can be expressed as a summation, yielding:

$$\mathbf{u}^T\boldsymbol{\Lambda}\mathbf{u} = \sum_i u_i{}^2\lambda_i. \tag{2.7}$$

Under this constrained problem, the maximum is achieved when $u_i{}^2 = 1$ with $u_j{}^2 = 0 \ \forall \ j \neq i$, with λ_i representing the maximum eigenvalue (q.e.d.).

The same argument can be extended to obtain subsequent principal components. As the eigenvectors of a symmetric matrix are orthogonal, the maximization problem of the second component is the same in Equation 2.7, restricted to the orthogonality constraint. The same applies for finding the k first components [19].

Assuming the eigenvectors are ordered in descending order with respect to the eigenvalues, being the $\mathbf{P}_k$ the projection matrix formed by the set of k-first eigenvectors, the data projection is obtained by:

$$\mathbf{G} = \mathbf{P}_k^T \mathbf{Y}. \tag{2.8}$$

Low-Rank Approximation

The computation of the covariance matrix and its spectral decomposition may be infeasible in practice for large matrices. A low-rank matrix approximation can be used to overcome this issue. Consider the singular-value-decomposition (SVD) of $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{2.9}$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices, and $\mathbf{\Sigma}$ a diagonal matrix containing the singular values, i.e. the root-square of the eigenvalues. Let include Equation 2.9 in Equation 2.1.

$$\mathbf{Q} = \mathbf{Y}\mathbf{Y}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T. \tag{2.10}$$

Comparing Equation 2.4 and Equation 2.10 it can be observed that $\mathbf{U}$ represents the matrix of eigenvectors $\mathbf{\Psi}$. Finally, if we substitute Equation 2.9 in Equation 2.8, and use the result of Equation 2.10 to substitute $\mathbf{\Psi}$ with $\mathbf{U}$, the projection of the data onto the eigenspace can be computed as:

$$\mathbf{G} = \mathbf{\Psi}^T\mathbf{X} = \mathbf{\Psi}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{\Sigma}\mathbf{V}^T. \tag{2.11}$$

Minimum Error Formulation

PCA has been defined via the covariance method. Nevertheless, the problem can be reformulated to find the minimum-error projection onto a lower-dimensional subspace of orthonormal basis vectors [20] (see Equation 2.12) under the Frobenius norm. Not surprisingly, the set of vectors that minimise 2.12 are the eigenvectors associated to the largest eigenvalues of the covariance matrix $\mathbf{Q} = \mathbf{Y}\mathbf{Y}^T$ [19] [20].

$$L(\mathbf{Y},\mathbf{P}) = \sum_i \|\mathbf{Y} - \mathbf{P}\mathbf{P}^T\mathbf{Y}\|_F^2 \tag{2.12}$$

It is shown in [20], that this formulation is equivalent as using the L_2 norm for the minimum-error projection. However, the introduction of this formulation provides another point of view of the problem, and allows to set other constraints such as the L_1 norm.

2.2.2 Two Dimensional PCA

Two dimensional PCA (2DPCA) [21] is an extension of PCA introduced by Yang et al. in the scope of face reconstruction and recognition. The idea behind this method is the same as standard PCA: find the optimal projection that maximizes the dispersion of the data, but in 2DPCA the data is arranged in matrices instead of in column vectors.

For a set of M not mean-centred data matrices $\hat{\mathbf{X}}_j$, the authors of 2DPCA define the measure of dispersion by the *inner-scatter matrix* $\mathbf{S}$:

$$\mathbf{S} = \frac{1}{M}\sum_{j=1}^{M}\left(\hat{\mathbf{X}}_j - \bar{\mathbf{X}}\right)^T\left(\hat{\mathbf{X}}_j - \bar{\mathbf{X}}\right), \tag{2.13}$$

where $\bar{\mathbf{X}}$ is the average matrix of the M matrices. For a given projection vector $\mathbf{v}_1$, the criterion J to maximize is:

$$J(\hat{\mathbf{X}}) = \mathbf{v}_1\mathbf{S}\mathbf{v}_1^T. \tag{2.14}$$

As in PCA, the optimal projections correspond to the eigenvectors of $\mathbf{S}$ associated to its largest eigenvalues. The relation between PCA and 2DPCA is that 2DPCA is essentially PCA performed on the rows of the data matrices, understanding each row as a data unit. In other words, 2DPCA of a data matrix can be seen as PCA on the set of rows of the matrix, where the covariance matrix is evaluated using the rows of all the centred training samples [22][23]. As a result 2DPCA has a smaller covariance matrix, as the dimension of each row is typically smaller than the dimension of each vectorized data.

2.2.3 2DPCA-L_1

The introduction of L_1-Norm-based 2DPCA [24] explored how to obtain basis vectors that maximize the data variance including the L_1 norm to provide robustness against outliers. The implicit square nature of L_2 norm of standard PCA (Section 2.12) entails that large differences in the J criterion are more penalized compared to the L_1 norm. Hence, the L_1 norm increases the method's robustness to outliers.

In [24], the problem of 2DPCA is re-formulated to find the set of vectors that maximize the dispersion of the projected data samples under the L_1 constraint. Let $\mathbf{X} = \mathbf{X_1}, \ldots, \mathbf{X_M}$ be a set of M mean-centred data matrices of size $m \times n$, and let $\mathbf{v}_1 \in \mathbb{R}^n$ be the first basis vector that maximizes the L_1-norm dispersion of the data. Then $\mathbf{v}_1$ can be obtained by maximizing:

$$Z(\mathbf{v}) = \sum_{i=1}^{M} \|\mathbf{X}_i \mathbf{v}\|_1, \tag{2.15}$$

$$\mathbf{v}_1 = \arg\max_{\mathbf{v}}(Z(\mathbf{v})), \tag{2.16}$$

constrained to

$$\|\mathbf{v}_1\|_2 = 1. \tag{2.17}$$

A more manageable expression can be obtained dividing Equation 2.15 into two summations. The product between each row of the matrix and the column vector can be explicitly separated by letting $\mathbf{x}_i^j$ represent the j-th row of the image $\mathbf{X}_i$:

$$Z(\mathbf{v}) = \sum_{i=1}^{M} \sum_{j=1}^{n} |\mathbf{x}_i^j \mathbf{v}|. \tag{2.18}$$

As the L_1 norm is not convex, its maximization must be carried out by an iterative approach. Let t represent the index of the iteration in the maximization process of a given principal component, the expression that iteratively maximizes $\mathbf{v}_1$ is:

$$\mathbf{v}_1(t+1) = \frac{\sum_{i=1}^{M} \sum_{j=1}^{n} s_{ji}(t) \mathbf{x}_i^j}{\|\sum_{i=1}^{M} \sum_{j=1}^{n} s_{ji}(t) \mathbf{x}_i^j\|_2}, \tag{2.19}$$

where $s_{ji}(t)$ is an auxiliary function that replaces the absolute value:

$$s_{ji}(t) = \begin{cases} 1, & if|\mathbf{x}_i^j \mathbf{v}_1(t)| > 0 \\ 0, & if|\mathbf{x}_i^j \mathbf{v}_1(t)| = 0 \\ -1, & if|\mathbf{x}_i^j \mathbf{v}_1(t)| < 0 \end{cases} \tag{2.20}$$

If the reader is interested, the proof of convergence of Equation 2.15 and extension to more principal components is described in [24].

2.2.4 Sparse 2DPCA-L_1

In [25], the authors propose to regularize Equation 2.18 to obtain sparsity (i.e. a subset of the input variables equal to zero) by means of elastic-net regularization [26]. Compared to standard PCA methods, where each principal component is a linear combination of all the input data features [26], the use of sparse methods encourages to reduce the number of features in the generation of the principal components, selecting a few salient features among all the input data, and allowing a more efficient encoding of the information.

The sparse 2DPCA-L_1 method is presented in Equation 2.21. Please note its similarity to Equation 2.18. However, two additional regularization constraints have been included. Their influence is controlled by the parameters η and γ. The first, $\|\mathbf{v}\|_2^2$, is known as the Tikhonov regularization, is a regularization term for optimization problems under the least square constraint that ensures a smooth numerical solution. The second term $\|\mathbf{v}\|_1$, usually referred as the *lasso* problem (least absolute shrinkage and selection operator) is included to enhance sparsity.

$$R(\mathbf{v}) = \sum_{i=1}^{M} \sum_{j=1}^{n} |\mathbf{x}_i^j \mathbf{v}| - \frac{\eta}{2}\|\mathbf{v}\|_2^2 - \gamma\|\mathbf{v}\|_1, \tag{2.21}$$

$$\mathbf{v}_1 = \arg\max_{\mathbf{v}}(R(\mathbf{v})). \tag{2.22}$$

An intuitive explanation of the effect of both terms can be done with the help of Figure 2.3. The Tikhonov regularization penalizes via the least square constraint inherent to L_2 norm, hence, it is not likely that small δ values get regularized and be set to zero, leaning towards not sparse solutions [27]. Differently, the regularization guided by the L_1 norm, penalizes proportionally large and small values, encouraging sparse solutions.

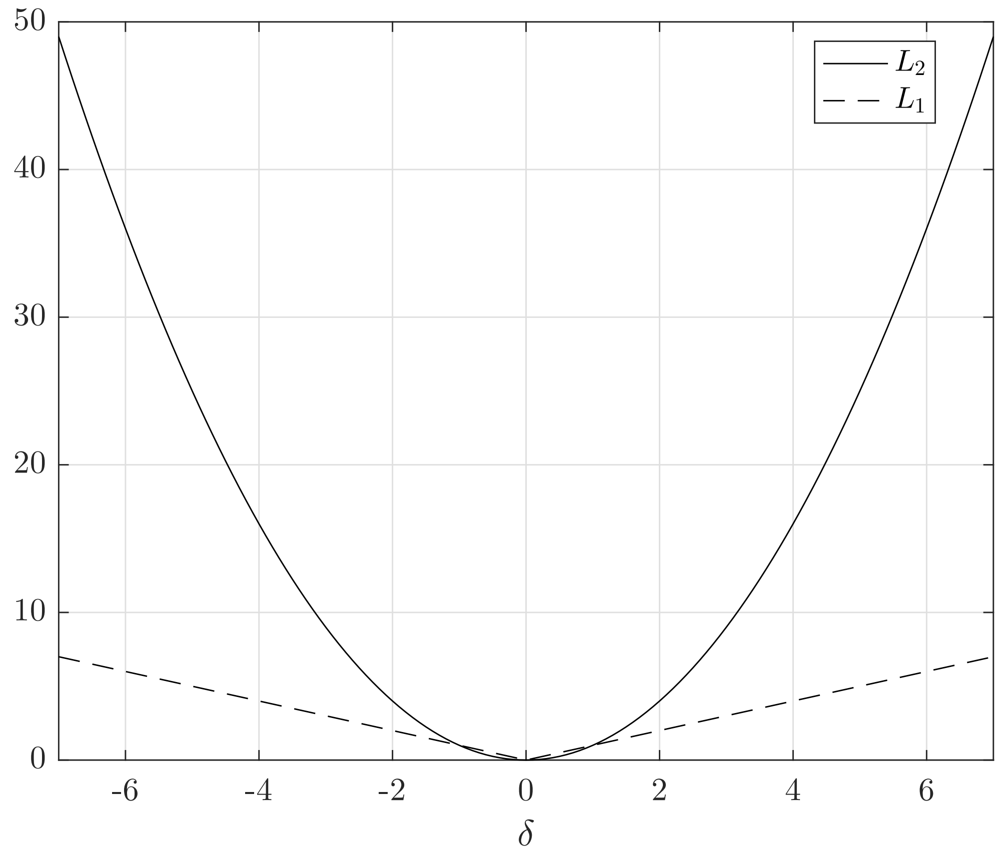

FIGURE 2.3: L_2 and L_1 penalization for a δ difference

The elastic-net combines both regularization methods. The L_1 constraint introduces sparsity, while the quadratic part of the penalty helps to stabilize the L_1 regularization path by removing the discontinuity.

Maximization Process

Given a set of M training matrices $\mathbf{X} = \mathbf{X}_1, \dots, \mathbf{X}_M$, with size $m \times n$, a number of desired basis vectors k, and regularization parameters η, γ, the procedure to obtain the k-set of principal components is described in [25] and included here, for the sake of completeness, in Algorithm 1.

Algorithm 1 Sparse 2DPCA-L_1 principal components computation

```
1: Let $\tau$ represent the index of the principal component, and initialize to $\tau = 1$.
2: Compute the principal component $\mathbf{v}_\tau$:
3:     Let $t$ be the index of the iteration. Set $t = 0$ and initialize $\mathbf{v}_\tau$ with any $n$ dimensional
       vector.
4:     Compute the value $s_{ji}(t)$ via Equation 2.20.
5:     Let $\mathbf{z}(t)$ represent the maximization expression for 2DPCA-$L_1$
           $$\mathbf{z}(t) = \sum_{i=1}^{M} \sum_{j=1}^{n} s_{ji}(t), \mathbf{x}_i^j \qquad (2.23)$$
       and $r(t)$ the expression that maximizes the regularization part
           $$\mathbf{r}(t) = \left( \frac{|v_1(t)|}{\gamma + \eta |v_1(t)|}, \dots, \frac{|v_n(t)|}{\gamma + \eta |v_n(t)|} \right), \qquad (2.24)$$
       where $v_p(t)$ is the entry with index $p$ of the vector that maximizes the dispersion of the
       data. The basis vector is maximized iteratively by letting
           $$\mathbf{v}_\tau(t+1) = z(t) \odot r(t), \qquad (2.25)$$
       where the symbol $\odot$ represents element-wise product.
6:     If the objective function $R(\mathbf{v}_\tau)$ does not grow more than a threshold $\epsilon$ with respect the
       last iteration, give as output $\mathbf{v}_\tau = \mathbf{v}_\tau(t+1)$. Otherwise set $t = t + 1$ and go to State 4.
           $$R(\mathbf{v}_\tau(t+1)) = \sum_{i=1}^{M} \sum_{j=1}^{n} |\mathbf{x}_i^j \mathbf{v}_\tau(t+1)| - \frac{\eta}{2} \|\mathbf{v}_\tau(t+1)\|_2^2 - \gamma \|\mathbf{v}_\tau(t+1)\|_1. \qquad (2.26)$$
7: Using the computed and normalized vectors $\mathbf{v_1} \dots \mathbf{v}_\tau$, the data is deflated by re-projecting
   the data rows as
           $$\mathbf{x}_i^j = \mathbf{x}_i^j - \sum_{l=1}^{\tau} \mathbf{v}_l (\mathbf{v}_l^T \mathbf{x}_i^j). \qquad (2.27)$$
8: If $\tau < k$ let $\tau = \tau + 1$ and return to State 2, where the deflated samples are used for finding
   the optimal vector $\mathbf{v}_{\tau+1}$. Otherwise stop and return the principal components
```

2.3 2D Image Representations from 3D Models

Proper rendering of 2D image representations from 3D models is a key step in codebook based approaches for pose estimation. If the method for sampling and the angle convention are not chosen properly, the representations may be biased towards a set of angles and may present comparison problems in the evaluation stage. There are three main conventions for representing rotations in the 3D space: rotation matrices, Euler angles and quaternions.

2.3.1 Rotation Matrices

Rotation matrices represent a rotation of θ degrees on the euclidean space by the product of a matrix with a vector. Expression 2.28 shows an example of the rotation matrix around the X axis of the standard basis in $\mathbb{R}^3$.

$$\Gamma_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin\theta \\ 0 & \sin(\theta) & \cos\theta \end{pmatrix} \tag{2.28}$$

Although their use may seem convenient, they suffer from numerical drift when implemented in finite-precision systems. The product of two orthonormal rotation matrices, due to floating-point errors may lose the orthonormality property, and the process for re-normalizing is usually ill-posed [28]. Additionally, is unclear how to define the distance function between two rotation matrices, as the matrix norms can be applied column-wise, row-wise or with respect to the eigen-values. This is inconvenient if metrics for evaluating the error of the algorithm are to be defined.

2.3.2 Euler Angles

The set of Euler angles on $\mathbb{R}^3$, can describe any orientation of a 3D object by rotating successively a set of three axes (u_1, u_2, u_3) with different rotations $(\theta_1, \theta_2, \theta_3)$. Euler angles have the benefit of being intuitive and numerically stable, but there are several problems associated to their use. First, different configuration of angles may lead to the same rotation, problem that is also manifested when two o more axes are aligned, this problem is known as gimbal-lock [28].

An example of gimbal-lock occurs in the configuration proposed in [3] for sampling all the possible orientations (represented in Figure 2.4). The object is rotated around the X and Z axes, whereas the rotation in Y is done by means of tilting the camera. In this configuration,

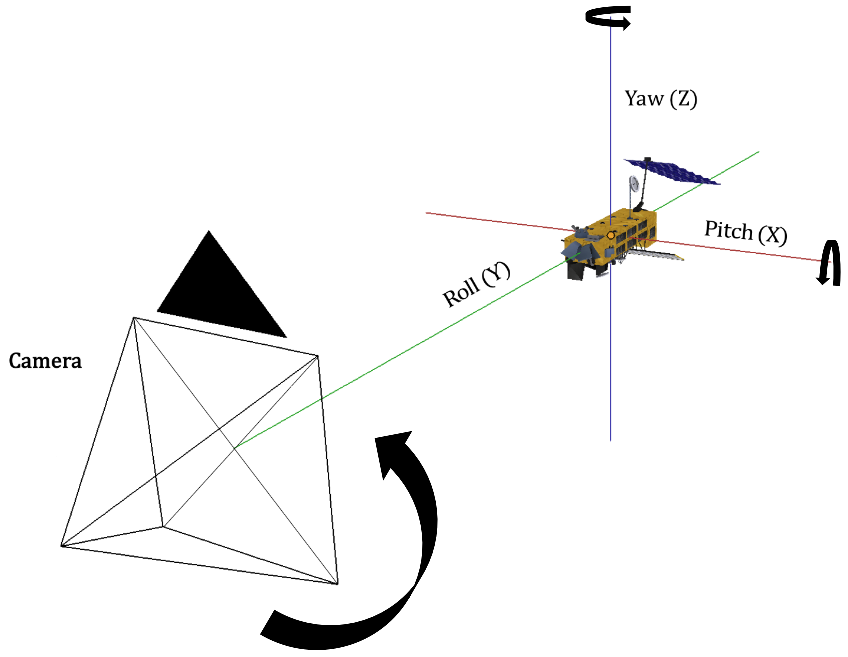

FIGURE 2.4: Attitude extraction method from [3]

when the X axis is perpendicular to the camera frame, the rotations in X are equivalent to the rotations in Y.

Another problem associated to the use of Euler angles is the use of distance metrics to evaluate the pose estimation. When comparing views close to the 360° warping or close to the gimbal-lock configuration, measuring distances between Euler angles is problematic, as two similar views may lead to a high error.

Finally, as shown in [29] [28], the uniform sampling of each independent Euler angle, does not provide an uniform sampling of the possible views of the 3D object. Differently, uniform sampling generates distributions heavily biased towards polar regions of the sphere according to the set of rotation axes [28].

The authors of [28] propose a method to overcome both the biased sampling and the distance problem derived from angle warping, by using random uniform distributions on the set of Euler angles and a proposed distance measure. However, the proposed distance is still unable to handle multiple representations of the same rotations, that occur close the gimbal-lock configuration described above. As in the proposed application the distance measure is critical to evaluate the goodness of the algorithm, the Euler angles do not provide a proper framework.

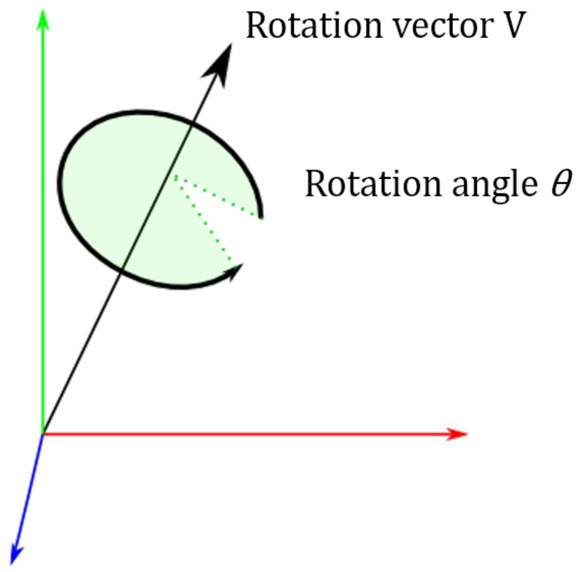

FIGURE 2.5: Quaternion representation as a vector + rotation

2.3.3 Quaternions

The third presented convention are the quaternions. They were formulated by William Hamilton in 1843, as a system to extend complex numbers to a four-dimensional vector. In a similar way in which multiplication by a complex number can be understood as a rotation in the two dimensions of the complex plane, quaternions define rotations in three dimensions, with the particularity that they need an extra dimension with respect to the space where the rotation is defined. A quaternion can be represented as $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$ with a, b, c, d real numbers and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ the fundamental quaternion units, subject to $\mathbf{i^2} = \mathbf{j^2} = \mathbf{k^2} = -1$. Please note that if $c = 0$ and $d = 0$ the quaternion expresses a complex number.

Each unit quaternion can be more intuitively understood as a rotation with the following reformulation:

$$Q = (w, x, y, x) = \left(\cos(\frac{\theta}{2}), v_x \sin(\frac{\theta}{2}), v_y \sin(\frac{\theta}{2}), v_z \sin(\frac{\theta}{2})\right), \tag{2.29}$$

where $V = (v_x, v_y, v_z)$ is a three dimensional vector, and θ a rotation around V. Figure 2.5 shows a representation of this concept, being the rotation axis defined by V and the rotation angle around V by θ.

As in Euler angles, quaternions present the problem associated to uniform sampling. If the sampling is not done following a random distribution, the estimates are biased and not uniformly distributed. In addition, two opposite quaternions, $Q = (w, x, y, x)$ and $-Q = (-w, -x, -y, -x)$, describe the same rotation. These problems can be overcome by defining a random sampling method based on uniform random distributions, and by defining a proper distance measure. The main advantage of using quaternions is that each rotation is described uniquely (with the exception of the two opposite quaternions), eliminating the problem of the gimbal-lock.

This makes quaternions the most suitable representation framework for defining the rotations, even though they do not provide an intuitive representation for rotations in the three dimensional space.

Chapter 3

Architecture

3.1 Introduction

The objective of the presented system is to estimate the pose of a target spacecraft, frame by frame, with a monocular camera placed on the chaser spacecraft. The target is assumed to be un-cooperative, placed at no more than 100m from the chaser, and in each image, the target must appear completely in the camera frame. In addition, the background is supposed to be mainly black with no other objects present in the scene. Conditions that are reasonable in space imagery, while neglecting some extension scenarios (e.g. Earth in the field of view) without loss of representativity.

In this master's thesis the pose is defined as:

- Attitude: As defined in Section 1.2.1, is the rotational orientation of a spacecraft in space with respect to a known frame.
- Position: Taking as reference the centre of mass of the target spacecraft, the position is defined as the location within the camera frame and the distance to the object.

The pose estimation is based on a codebook approach, where each input image is compared with the entries of a codebook with image representations associated to poses. The pose is retrieved from the entry of the codebook with the minimum distance match to the input image.

Figure 3.1 shows a block diagram of the proposed system, divided in two parts: codebook training and classification. In the training part, given a 3D model of the spacecraft, reference

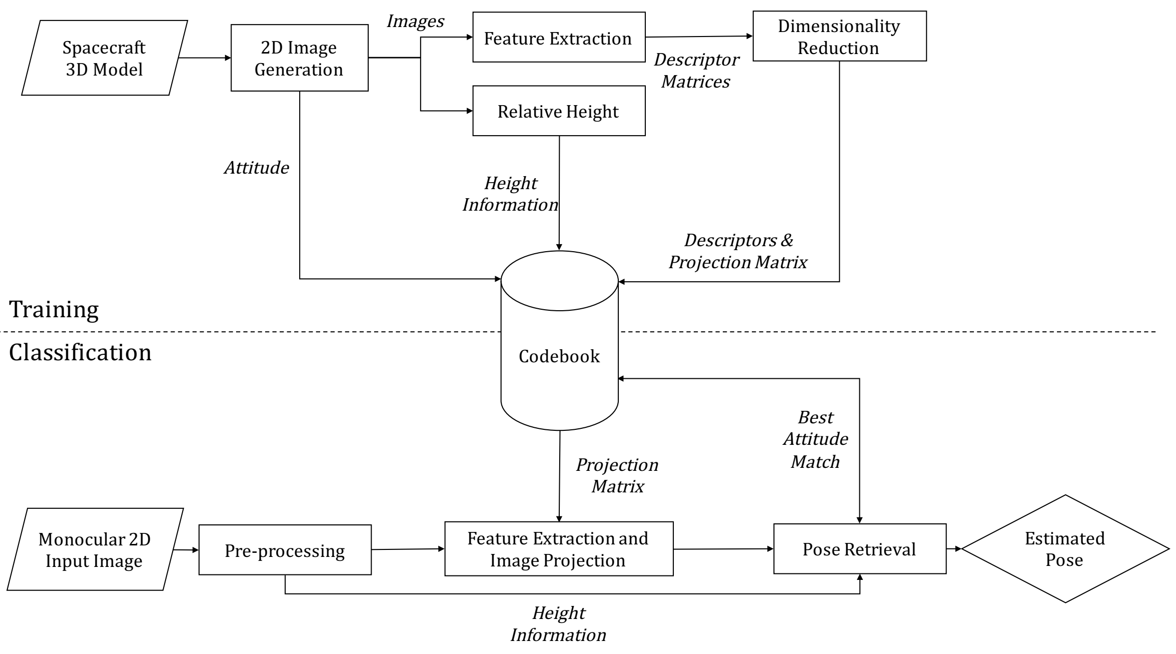

FIGURE 3.1: Architecture of the proposed system for spacecraft pose estimation

images are extracted by randomly sampling on the full set of attitudes. In the pre-processing stage, the image is normalized by the energy and the spacecraft is cropped from the image to achieve scale invariance. This cropped image, resized to a common size, is described by means of the pixel intensity values and Histograms of Oriented Gradients (HOG)[30] (see Appendix A). Finally, a smaller representation of the descriptors, obtained with the sparse 2DPCA-L_1 algorithm is used to build the codebook.

The classification part follows a similar procedure, each input image is processed to generate an image containing only the target spacecraft. Then, the image is described by means of the normalized intensity values and HOG descriptor. The descriptor is projected onto the principal components obtained in the training stage, to retrieve the attitude by finding the nearest-neighbour in the codebook. Finally the position of the spacecraft is given, along with the distance to the object which is computed using information associated to the pose.

3.2 Training

3.2.1 2D Image Generation

The first step to generate the codebook is to obtain a representative 3D-model of the target spacecraft. For this master's thesis, a publicly available model of the Envisat created by Grant Borodin [31] is used and reproduced under his permission.

To render the images, the open-source Blender 3D creation suite has been used [32]. This suite allows to simulate different lighting conditions along with a Python interface, which provides enough flexibility for the required application.

Attitude Sampling

The aim of this stage is to create a subset of images, corresponding to attitudes, that best describe the all possible three-dimensional orientations of the target spacecraft.

It was seen in Section 2.3 that methods relying on uniform sampling of the angle representation space, based on Euler angles or quaternions, provide a biased estimate of the attitude (i.e. the set of orientations is not uniformly distributed). Additionally, the use of Euler angles may pose a problem if distance metrics are going to be used for evaluation, due to the gimbal-lock configuration.

The quaternions provide a good framework for evaluation, as they do not suffer from gimbal-lock issues. Moreover, if the set of quaternions representing the set of possible orientations is sampled with a random uniform distribution, the resulting subset is uniform [28]. Kuffner provides an algorithm in [28] to generate uniformly distributed random quaternions, that relies on the random sampling of the possible vectors and orientations that describe a quaternion.

Assuming $rand()$ is a function that generates pseudo random numbers from the uniform distribution between $[0, 1]$, the set of quaternions $Q = (w, x, y, x) = \big(\cos(\frac{\theta}{2}), v_x \sin(\frac{\theta}{2}), v_y \sin(\frac{\theta}{2}), v_z \sin(\frac{\theta}{2})\big)$, described by a rotation angle θ and a three-dimensional vector (x, y, z), can be uniformly with the procedure shown in Algorithm 2.

Algorithm 2 Sampling of uniformly distributed random quaternions

```
1: Define a number of random orientations N
2: for each orientation do
3:     Get a sample from the uniform random distribution $s = rand()$
4:     Define $\sigma_1 = \sqrt{1-s}$ and $\sigma_2 = \sqrt{s}$
5:     Randomly generate two rotations by sampling the random distribution again
6:         $\theta_1 = 2\pi \times rand()$
7:         $\theta_2 = 2\pi \times rand()$
8:     Compute the quaternion values as
9:         $w = \cos(\theta_2) \times \sigma_2$
10:        $x = \sin(\theta_1) \times \sigma_1$
11:        $y = \cos(\theta_1) \times \sigma_1$
12:        $z = \sin(\theta_2) \times \sigma_2$
13:    Set $Q_i = (w, x, y, x)$
14: end for
```

3.2.2 Relative Height Computation

This module has as input the images corresponding to attitudes generated by the procedure described in Section 3.2.1, and gives as output the relative heights corresponding to each attitude.

The relative height is understood as the height that the spacecraft presents in a given attitude, represented by r in Figure 3.2. The computation of this parameter is necessary to estimate the distance to the object in the classification stage, as the estimation of the distance to an object with a monocular camera can only be achieved if the relative height of the object is known beforehand.

To justify the computation of this parameter, the development to obtain the distance to the object will be presented here, even though it is used in the classification stage. In this development, a simplified camera pinhole model is used, then, the relationship between an object in the three-dimensional space, and its projection onto the camera plane can be described by a triangle similarity, as shown in Figure 3.2,

where:

- d: represents the distance from the camera pinhole to the target in mm.
- f: is the distance from the pinhole to the image plane in mm.
- r: is the relative height of the object for a given attitude in mm.
- o: is the observed height of the object projected in the sensor, expressed in mm.

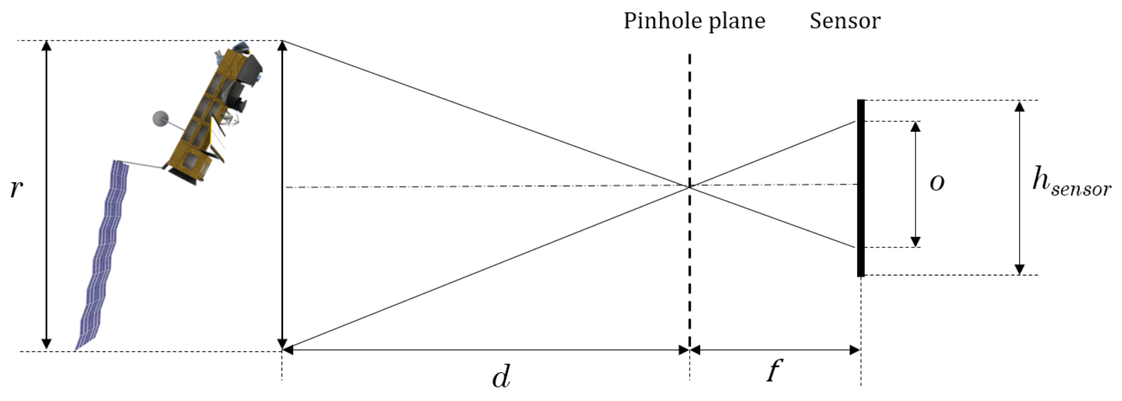

FIGURE 3.2: Simplified schema of an object projection in the sensor

- h_{sensor}: the height of the sensor in mm.

The triangle similarity is expressed by,

$$\frac{o(mm)}{f(mm)} = \frac{r(mm)}{d(mm)} \tag{3.1}$$

$$d(mm) = \frac{f(mm) \times r(mm)}{o(mm)}. \tag{3.2}$$

The observed height o of the projected object in the sensor can be further expanded to relate it with the number of pixels that occupies in the image, by knowing the size of the sensor $h_{sensor}(mm)$, the height of the image $h_{image}(px)$, and the height of the object in the image as a measure of pixels $h_{object}(px)$, yielding

$$d(mm) = \frac{f(mm) \times r(mm) \times h_{image}(px)}{h_{sensor}(mm) \times h_{object}(px)}. \tag{3.3}$$

Please note that for estimating the distance it is necessary to know the relative height $r(mm)$ of the target associated to each attitude. This parameter is computed offline in this stage and associated to each attitude in the codebook. Unfortunately, the used 3D environment does not allow to compute the relative height in a given orientation. To overcome this issue, as the distance between the camera and the object is known in the 3D environment, the equation is rearranged to obtain the relative height associated to each attitude in the training stage, where parameter $h_{object}(px)$ is computed as the maximum number of pixels belonging to spacecraft in the vertical direction:

$$r(mm) = \frac{h_{sensor}(mm) \times h_{object}(px) \times d(mm)}{f(mm) \times h_{image}(px)}. \tag{3.4}$$

3.2.3 Feature Extraction

This stage extracts the features from the generated images prior to its dimensionality reduction via 2DPCA-L_1 with sparsity presented in Section 2.2.4

The dataset with M images corresponding to sampled attitudes, is described by the concatenation of the pixel intensity values of the grey-scale image, and Histogram of Oriented Gradients descriptors projected over the optimal basis vectors found with sparse 2DPCA-L_1.

For a given image with index $j = 1, \ldots, M$, let $\mathbf{I}_j$ denote the pixel intensity descriptor, and $\mathbf{H}_j$ the HOG descriptor.

The process to obtain $\mathbf{I}_j$ and $\mathbf{H}_j$, starts by labelling the pixels representing the spacecraft of the training images. This can be done by a simple thresholding and closing morphological operators, as the background does not contain noise.

To achieve scale invariance, the spacecraft is extracted by cropping the image to the minimum-area rectangle circumscribing the labelled pixels. From this step, the acquisition of $\mathbf{I}_j$ and $\mathbf{H}_j$ follows different procedures:

- To obtain $\mathbf{I}_j$, the image $\hat{\mathbf{I}}$ is resized to a common $m \times n$ size and normalized by the energy $\mathbf{I}_j = \hat{\mathbf{I}}/\bar{\mathbf{I}}$ to mitigate the variations in illumination. The normalization factor $\bar{\mathbf{I}}$ is computed as

$$\bar{\mathbf{I}} = \left[\sum_{i=1}^{m} \sum_{j=1}^{n} \hat{\mathbf{I}}_{i,j}^{2} \right]^{1/2}. \tag{3.5}$$

- $\mathbf{H}_j$ is obtained by computing the HOG descriptors over a resized image of size $m' \times n'$, where $m' < m$ and $n' < n$. This prevents the HOG descriptor from be too large, which maybe infeasible for real spacecraft applications. To allow the concatenation of descriptors in a matrix form, the HOG descriptor is resized with linear interpolation to a matrix with size $m \times c$.

Then, both descriptors are concatenated to conform the final descriptor $\mathbf{D}_j$ of size $m \times (c+n)$, where the j represents the index $j = 1, \ldots, M$ of the image from the training set with M elements.

$$\mathbf{D}_j = \begin{bmatrix} \mathbf{I}_j & \mathbf{H}_j \end{bmatrix}. \tag{3.6}$$

3.2.4 Dimensionality Reduction

Once the whole set of descriptors is computed, the dimensionality is reduced through the projection onto the principal components computed with sparse 2DPCA-L_1. First, to force the basis vectors to pass through the origin and make the data independent of the mean value, the descriptor matrix is mean centred across the third dimension, as done in 2DPCA (Section 2.2.2).

In the proposed application, the initialization of the sparse 2DPCA-L_1 basis vectors is done with the basis vectors of 2DPCA, this is, the eigenvectors associated to the *inner-scatter matrix*, with $\bar{\mathbf{D}}$ representing the mean across descriptors:

$$\mathbf{S} = \frac{1}{M} \sum_{j=1}^{M} \left(\mathbf{D}_j - \bar{\mathbf{D}}\right)^T \left(\mathbf{D}_j - \bar{\mathbf{D}}\right). \tag{3.7}$$

The set of principal components are computed following the procedure described in Algorithm 1 from Section 2.2.4. The final descriptor $\mathbf{D}_j^P$ is built by projecting the data over the first k principal components spanned by the basis vectors $\mathbf{P_k}$:

$$\mathbf{D}_j^P = \mathbf{D}_j \mathbf{P_k}. \tag{3.8}$$

The projected descriptors associated to each pose are stored to the codebook. In addition, the projection matrix $\mathbf{P_k}$ is also stored, as it is necessary to project the new input descriptors form the classification stage.

3.2.5 Codebook Generation

The final step of the training stage is to generate the codebook. The proposed structure is presented in Table 3.1. The pose fields contain the descriptor matrices $\mathbf{D}_j^P$ associated to each attitude described by a quaternion Q_j. The height field r_j, associated to the attitude, contains the relative height of the target spacecraft for a given orientation. This will allow to have an approximate estimate of the distance to the object with the use of a monocular camera.

Pose			
Index	Quaternion	Relative Height (mm)	Descriptor
1	Q_1	r_1	$\mathbf{D}_1^P$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
M	Q_M	r_M	$\mathbf{D}_M^P$

Table 3.1: Proposed codebook structure

3.3 Classification

3.3.1 Image Pre-processing

The input of this stage is an image from the camera sensor containing the target spacecraft. The outputs are the pre-processed image needed for the descriptor extraction, and the $h_{object}(px)$ parameter needed for the distance estimation.

The goal of this stage is to segment the pixels representing the target spacecraft from the background, which can be modelled as a set of black pixels with small noise from the input sensor.

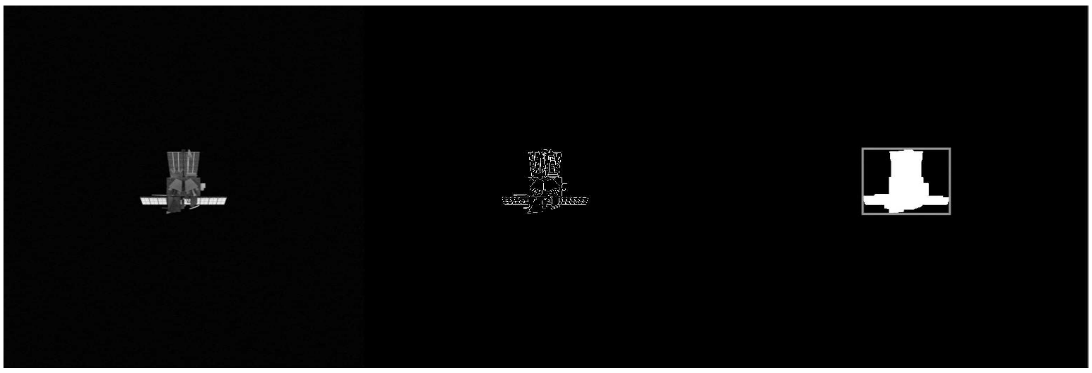

Figure 3.3: Example of spacecraft detection and cropping

The process of detecting the spacecraft starts with the application of a Gaussian filter to average the noise coming from the input sensor. The object is detected by the application of a Sobel edge detector, as it is supposed to be in a homogeneous background. A closing morphological operator is used to fill the contours of the edges prior the application of a blob colouring algorithm to detect the different connected components. Finally, the blob with the highest number of pixels is assumed to be the spacecraft. This process is illustrated in Figure 3.3.

The maximum number of pixels belonging to spacecraft in the vertical direction, is given as the relative height of the object in pixels for the pose retrieval stage. The cropped image containing the spacecraft is given as input for the Feature Extraction and Image Projection stage.

3.3.2 Feature Extraction and Image Projection

The features describing the input image from the pre-processing stage are extracted by following the same procedure described in Section 3.2.3. The new obtained descriptor is projected onto the projection matrix conformed by the principal components evaluated in Section 3.2.4.

3.3.3 Pose retrieval

The projected descriptor is used to retrieve the attitude of the spacecraft by finding the nearest-neighbour in the codebook under the L_1 norm. This distance metric proved to be the most discriminative under preliminary tests.

Once the attitude is retrieved, the distance from the camera to the object is computed with Equation 3.3, using the relative height associated to the pose, and the relative height in pixels obtained from the pre-processing stage. Finally, the centre of the bounding box containing the spacecraft is given as position in the camera frame.

Chapter 4

Tests and Results

The aim of this chapter is to present an objective evaluation of the proposed method for spacecraft pose estimation. First, the evaluation metrics and experimental data are presented, to continue with the different experiments and finish with the discussion of the results.

The experiments are compared with the approach proposed in [3]. Although the authors kindly provided the code for this master's thesis, their implementation did not allow to use a large number of training images due to memory requirements. For this reason, their approach was also implemented in MATLAB.

4.1 Evaluation Metrics And Experimental Data

4.1.1 Evaluation Metrics

The evaluation metrics are based on the quaternion distance measure presented in [28] and the computational time. The distance metric relies on the dot product between quaternions, taking into account that polar opposite quaternions, namely $Q = (w, x, y, x)$ and its opposite $-Q = (-w, -x, -y, -x)$ represent the same rotation. The distance, expressed by $dist(Q_1, Q_2)$, is computed as:

$$dist(Q_1, Q_2) = 1 - \|\mu\|, \tag{4.1}$$

where

$$\mu = Q_1 \cdot Q_2. \tag{4.2}$$

Two polar opposite quaternions give the same distance thanks to the use of $\|\mu\|$, as this occurs when $\mu = Q_1 \cdot Q_2$ and $-\mu = -Q_1 \cdot Q_2 = Q_1 \cdot -Q_2$. The resulting distance is between 0 and 1, with 0 representing two equal unit quaternions. The quaternion distance is used to define two error metrics:

- Mean Quaternion Error (MQE): is the quaternion distance between the attitude retrieved in the codebook and the real attitude from the test image, averaged over the total number of test images. Let N be the total number of images in the test set, Q_{cd} the quaternion retrieved from the codebook and Q_i the quaternion describing the attitude from the test image with index i.
$$MQE = \frac{1}{N}\sum_{i=1}^{N} dist(Q_{cd}, Q_i). \tag{4.3}$$
- Percentage of distances over a threshold $dist_{\%}$. It is used to give an insight of the number of images that lead in a high error estimate.

$$dist_{\%} = 100 \times \frac{\#\text{ of }(dist(Q_{cd}, Q_i) > threshold)}{N}, i = 1, \ldots, N. \tag{4.4}$$

The computational time is measured using a MATLAB implementation of the algorithm accounting the whole image processing pipeline in test, i.e. the time lapse ranging from the image capture to the pose detection. The implementation runs on MATLAB 2014b in a computer with 8Gb of RAM, an AMD FX-8320 processor (2012) and Windows 10.

4.1.2 Datasets

The set of attitudes used to generate the training dataset are obtained independently from the ones used to generate the test datasets. This is done to measure the capability of the codebook in a realistic situation where the attitudes will not exactly match, as the codebook contains a randomly sampled subset of the all possible attitudes.

A summary of the main characteristics of the different datasets is presented in Table 4.1, representative image examples in each dataset are shown in the sections describing the associated experiments.

Name	**N° of images**	**Distance from camera to target**	**Illumination conditions**	**Sampling strategy**
Training	1000-10000	50m	Even global illumination	Randomly generated attitude
Equal Cond	3000	50m	Even global illumination	Randomly generated attitudes
Diff Light 1	1000	50m	Even direct illumination	Randomly generated attitudes
Diff Light 2	500	50m	Lateral soft illumination	Randomly generated attitudes
Diff Light 3	500	50m	Lateral hard illumination	Randomly generated attitudes
Different Dist 1	500	75m	Lateral soft illumination	Same attitudes in *Diff Light 2*
Different Dist 2	500	100m	Lateral hard illumination	Same attitudes in *Diff Light 3*

Table 4.1: Summary of the datasets used in the experiments.

4.1.3 Reference Metrics

To provide an insight of the coarseness of the partition obtained in the training set, Table 4.2 relates the number of generated attitudes, with the mean minimum quaternion distance. To obtain this metric, the minimum distance for each quaternion is retrieved by comparing each quaternion against the rest (except with itself). Then, the minimum distances are averaged to compute the value present in Table 4.2. This coarseness measure may help to assess the MQE obtained in the experiments, i.e. it represents a sort of a step or bin width to define tolerable quaternion errors.

In addition, to relate the mean minimum quaternion error with a more intuitive metric, the correspondences between the number of generated images and the Euler angle step necessary to cover all the possible attitudes is presented. The generation is done following the method described in [3], where the angles corresponding to the gimbal-lock are removed. A visual cue for the differences in Euler angles is given in Figure 4.1, where the visual differences resulting from the sweeping of Euler angles over the Z axis are shown.

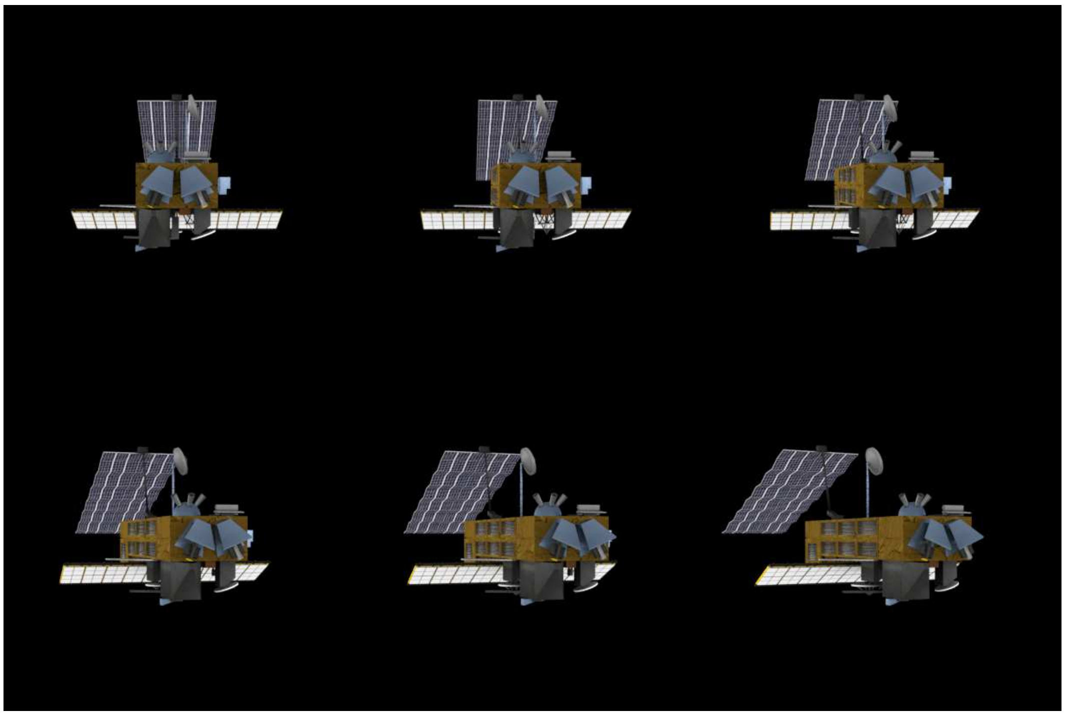

Figure 4.1: Example of angle sampling on the training dataset, over a rotation defined in the Z axis. First row, from left to right: 0°, 5°, 10°. Second row, from left to right: 15°, 20°, 27°.

Number Of Images	**Quaternion Coarseness**	**Angle Sampling**
1,000	0.0068	27°
3,000	0.0036	20°
6,000	0.0024	15°
10,000	0.0006	13°

Table 4.2: Correspondence between the number of randomly sampled attitudes, step in Euler angles and quaternion coarseness.

The input images have a size of 1024×1024 pixels. They are resized to 264×264 pixels to generate the intensity descriptors and to 128×128 pixels for the generation of the HOG descriptors with a bin size of 4 and 9 orientations. The final descriptor before projection resulting from the concatenation of the intensity values and the resized HOG descriptors (see Section 3.2.3), has a size of 264×528 which is reduced by projecting onto the principal components to $528 \times k$, where k represents the number of principal components.

4.2 Experiments

This section presents the different experiments and results with graphs to show the evolution of the error metrics depending on different parameters. During the experiments only minor comments are done to leave free interpretation to the reader. At the end of this section, a summary table is presented and a detailed discussion is done with the help of the table.

4.2.1 Experiment 1: Optimal Codebook Size

Number of Training Images

The first experiment is designed to choose the optimal number of training images and principal components, by evaluating how the size of the dataset affects the MQE and computational time. The optimal size is chosen as a trade-off between MQE and computational time. In order to reduce the heft of illumination conditions and target distance, this experiment is carried out on the *Equal Cond.* dataset.

For this purpose, the largest available *Training* set composed of 10,000 images corresponding to random attitudes, is randomly sampled to generate four different codebooks of 1,000, 3,000, 6,000, and 10,000 entries, with a fixed number of $k = 50$ principal components. This number is chosen in line with the studies [25] [3], where in their respective applications, a set of $k = 30$ components was enough to encode the information. A larger number of principal components is chosen here as a conservative solution to account for a bigger dataset.

The mean processing time per image and the MQE, both obtained when detecting attitudes of image samples in the *Equal Cond.* dataset by searching on each of the four generated codebooks as a function of the number of training images, are presented in Figure 4.2. Although the minimal MQE is achieved with the dataset containing 10,000 images, its has been considered that a good trade-off between MQE and computational time is obtained with the training dataset of 6,000 images.

Number of Principal Components

To assess the impact of the number of principal components in terms of MQE and computational cost, the same experiment is performed but as a function of the number of components with the codebook built with 6,000 entries. The results of this experiment are shown in Figure 4.3. The best trade-off between computational time and MQE is considered to be achieved with $k = 30$

principal components. Hence, a codebook size of 6,000 attitudes with 30 principal components is chosen, and used in the following experiments. The same number of images and principal components is used for the PCA algorithm.

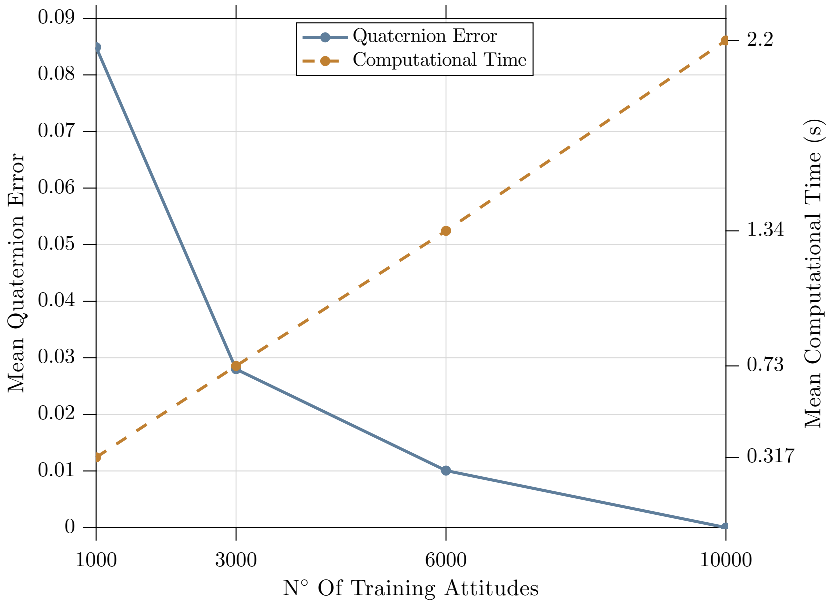

FIGURE 4.2: Mean processing time for one image, and *MQE* over *Equal Cond.* as a function of the number of training images with 50 principal components

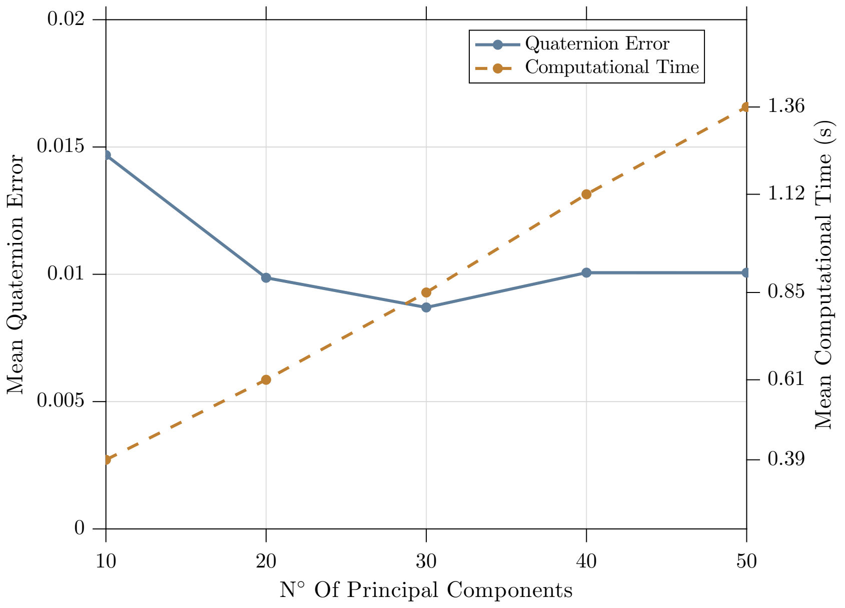

FIGURE 4.3: Mean processing time for one image, and *MQE* over *Equal Cond.* as a function of the number of principal components with 6,000 training images

4.2.2 Experiment 2: Invariance to Illumination Conditions

This test is designed to measure the performance of the algorithm in different illumination conditions. The codebook trained with 6,000 images and 30 principal components is tested over the datasets *Diff Light 1*, *Diff Light 2*, and *Diff Light 3*, with the respective examples shown in Figure 4.4.

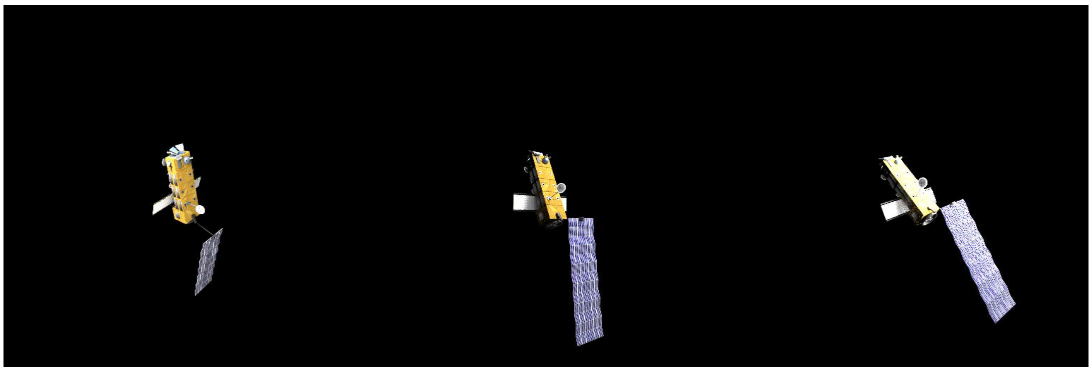

FIGURE 4.4: Different simulated illumination conditions from the test datsets. From left to right: *Diff Light 1*, *Diff Light 2* and *Diff Light 3*.

From this experiment, until the end of the chapter, detection results are evaluated via the percentage of distances over a threshold $dist_{\%}$, as a function of the threshold. The threshold value is chosen to start in the mean minimum quaternion error for 6,000 attitudes presented in Table 4.2. This value δ, that can be considered as the quaternion step between attitudes, is sequentially increased by a factor of 1 until 30 in steps of 0.25. The final threshold, with a value of 30δ, corresponds to an angle difference of $30°$ in each one of the Euler angles, which is considered as the usability limit of the algorithm.

Figure 4.5 presents the results when detecting attitudes in the *Diff Light 1* dataset. The results corresponding to *Diff Light 2* and *Diff Light 3* are presented in Figure 4.6, and Figure 4.7 respectively.

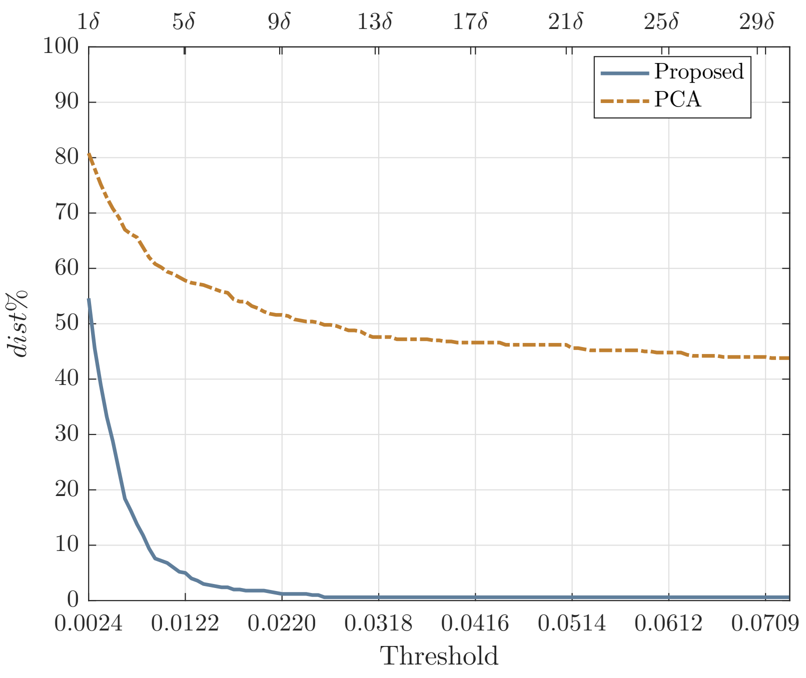

FIGURE 4.5: Percentage of error above a threshold as a function of the threshold ($\delta = 0.0024$). Tested over *Diff Light 1*. Codebook built with 6,000 attitudes and 30 principal components.

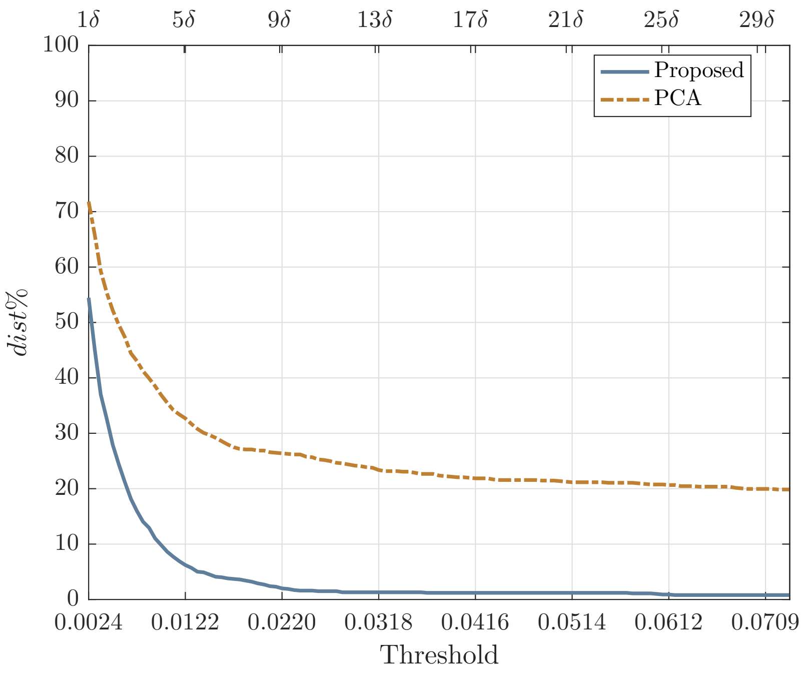

FIGURE 4.6: Percentage of error above a threshold as a function of the threshold ($\delta = 0.0024$). Tested over *Diff Light 2*. Codebook built with 6,000 attitudes and 30 principal components.

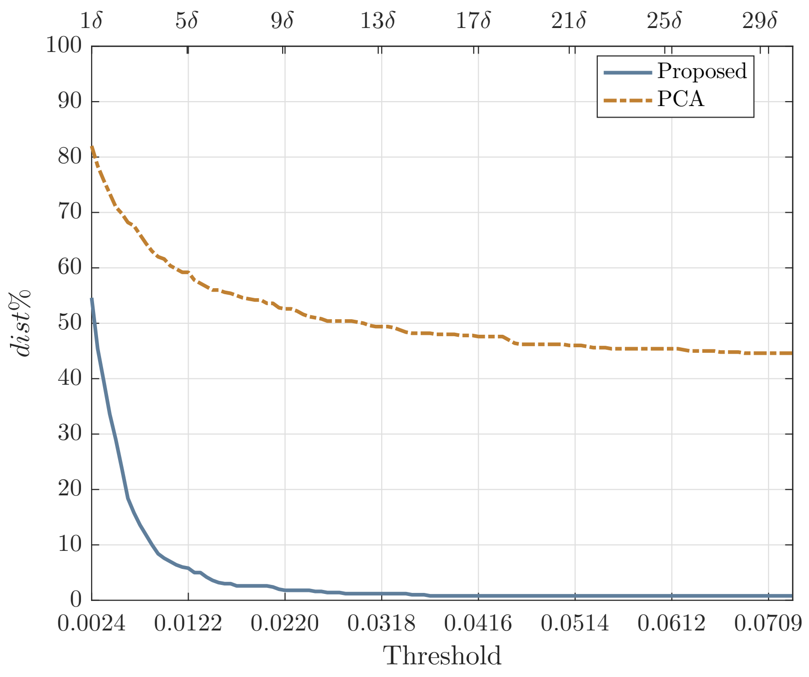

Figure 4.7: Percentage of error above a threshold as a function of the threshold ($\delta = 0.0024$). Tested over *Diff Light 3*. Codebook built with 6,000 attitudes and 30 principal components.

4.2.3 Experiment 3: Invariance to Scale and Distance Estimation

Invariance To Scale

The Experiment 3 is designed to test how the algorithm behaves when the target is placed at different distances from the chaser, with respect to the one considered when building the codebook. Having this in mind, the $dist_{\%}$ measure as a function of the threshold is obtained here when analysing images in the datasets *Different Dist 1* (distance from the camera to the object 75m, results in Figure 4.9) and *Different Dist 2* (distance from the camera to the object 100m, results in Figure 4.10).

Distance Estimation

This test presents the experiments to measure the distance from the camera to the spacecraft. For this purpose, the relative height r of the spacecraft in each attitude from the training dataset is computed via Equation 3.4. Then, for each retrieved attitude, the distance is computed with Equation 3.3 using the precomputed relative height r. Figure 4.11 and Figure 4.12 show the distance estimates for the *Different Dist 1* and *Different Dist 2*, with the target captured at 75 and 100 meters respectively.

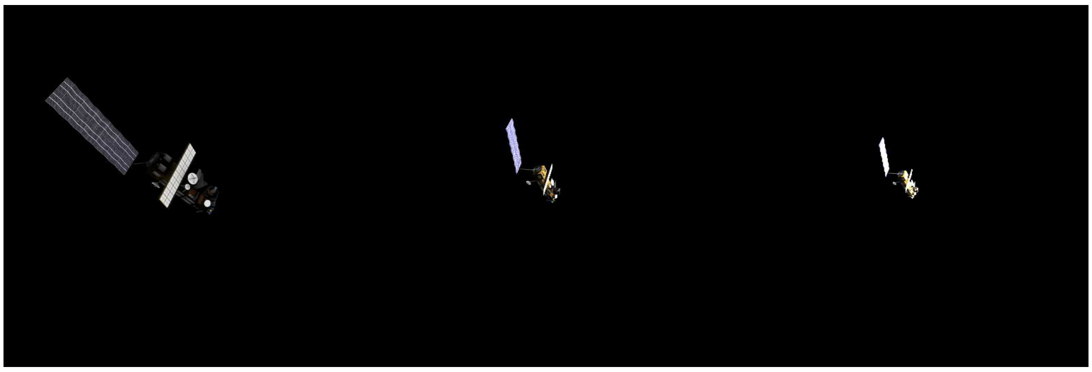

FIGURE 4.8: Comparison of the different scales. From left to right: Testing dataset, *Different Dist 1* at 75 meters, *Different Dist 2* at 100 meters

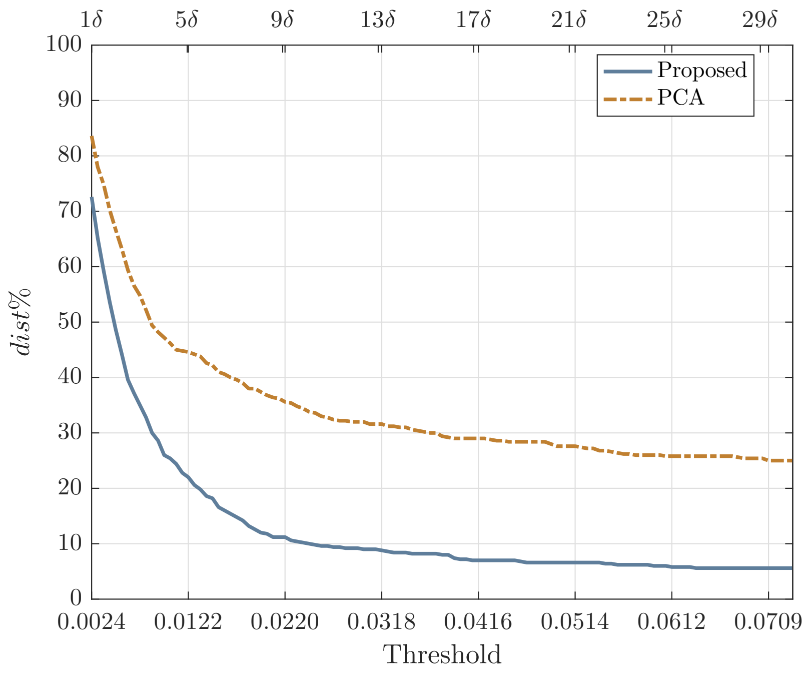

FIGURE 4.9: Percentage of error above a threshold as a function of the threshold ($\delta = 0.0024$). Tested over *Different Dist 1* representing a distance of 75 meters from the target. Codebook built with 6,000 attitudes and 30 principal components.

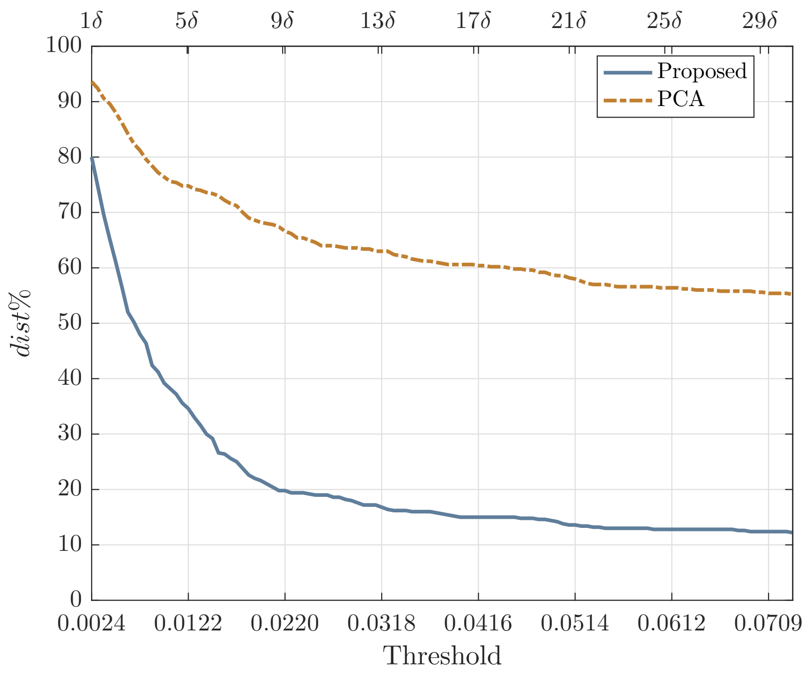

FIGURE 4.10: Percentage of error above a threshold as a function of the threshold ($\delta = 0.0024$). Tested over *Different Dist 2* representing a distance of 100 meters from the target. Codebook built with 6,000 attitudes and 30 principal components.

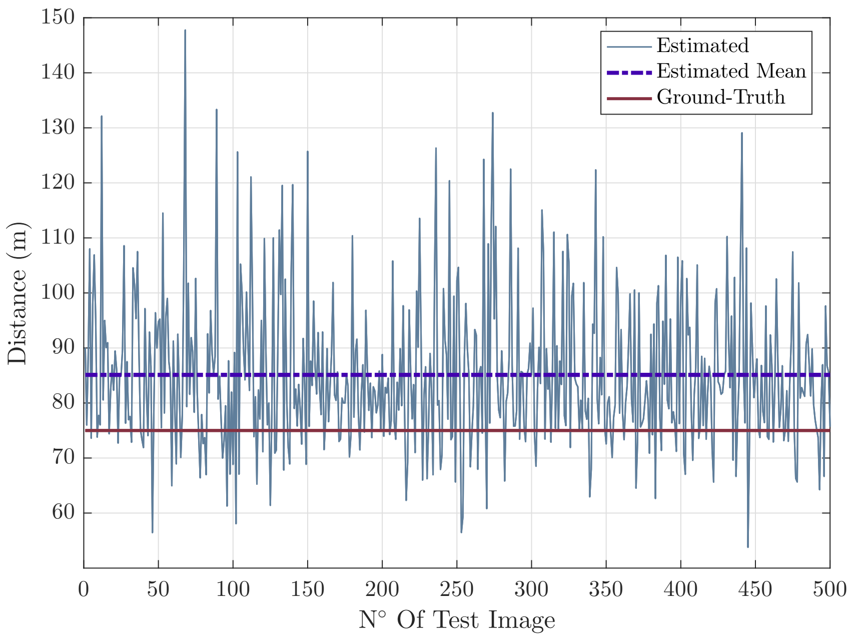

FIGURE 4.11: Distance estimation at 75 meters.

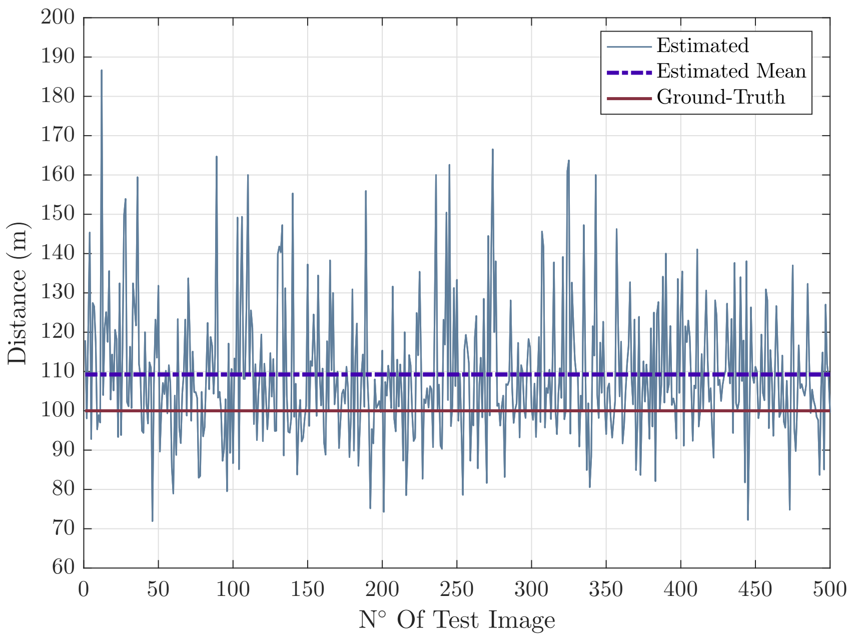

FIGURE 4.12: Distance estimation at 100 meters.

4.2.4 Experiment 4: Invariance to Noise

This experiment aims to measure how the algorithm behaves under two different types of noise that may appear in the case of study: blur motion and Gaussian noise. On one hand, the motion blur may arise in situations where the target is rotating fast, and the low illumination condition does not allow to capture with fast exposure times. On the other hand, the Gaussian noise can appear as result of the overheating of the camera sensor or other components in the chaser spacecraft such as the amplifiers. For this test, Gaussian noise with zero mean and variance equal to 0.005 has been added to the dataset *Diff Light 2*. The motion blur, with 25 pixels of displacement, is added to *Diff Light 1*.

Figure 4.13 shows an example of each image. The results are presented in Figure 4.13 and Figure 4.15, for motion blur and Gaussian noise respectively.

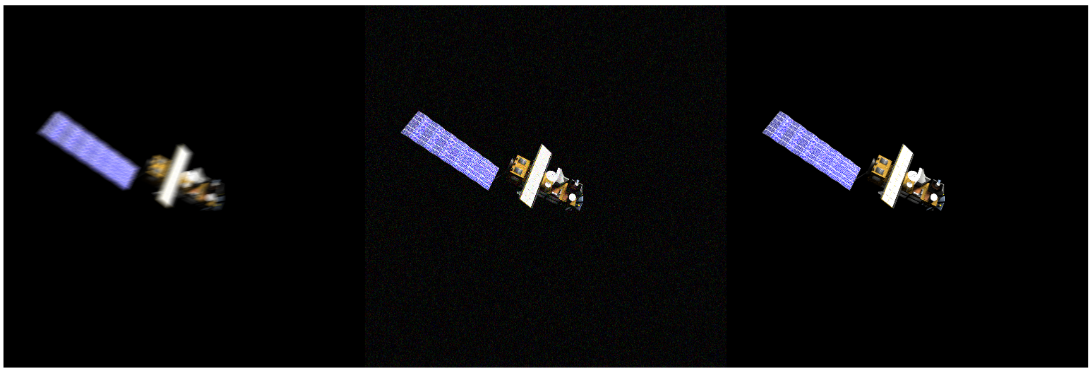

Figure 4.13: Example of noise motion blur (left), Gaussian noise (centre), and original image (right)

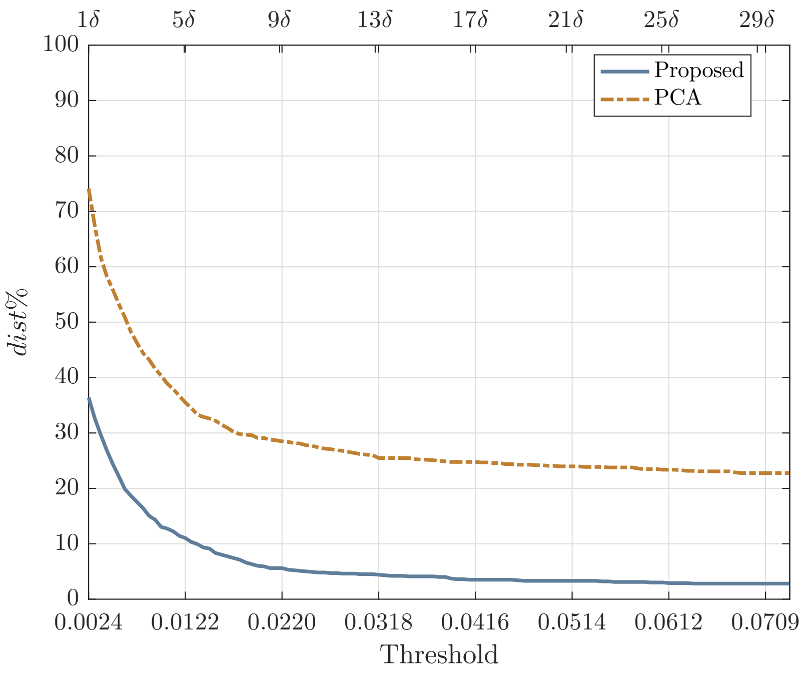

Figure 4.14: Percentage of error above a threshold as a function of the threshold ($\delta = 0.0024$). Tested over *Diff Light 2* with motion blur. Codebook built with 6,000 attitudes and 30 principal components.

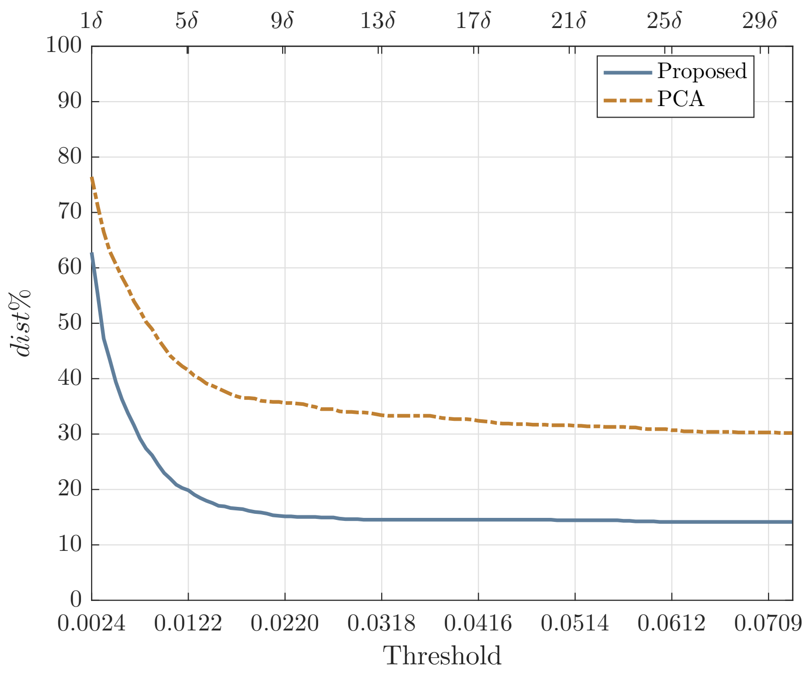

FIGURE 4.15: Percentage of error above a threshold as a function of the threshold ($\delta = 0.0024$). Tested over *Diff Light 3* with Gaussian noise. Codebook built with 6,000 attitudes and 30 principal components.

4.3 Summary and Discussion

A summary of the MQE measure obtained when detecting attitudes in the different datasets is presented in Table 4.3. The Δ *error* column includes the factor of MQE increment between results obtained for the dataset in the row and a reference dataset specified in brackets, aiming to compare results between datasets with similar characteristics. Δ *error* is obtained by:

$$\Delta error = \frac{MQE}{MQE_{ref}} \tag{4.5}$$

For instance, in Experiment 2, the datasets are similar to the one used in Experiment 1 but including variations in illuminations and attitudes, hence, Δ *error* represents here the factor of error increment derived from the illumination changes. Similarly, Experiments 3 and 4 are carried out on variations of the *Diff Light 2* and *Diff Light 3* datasets used in Experiment 2, changing the distance or and adding noise respectively. Therefore, as the illumination and attitudes remain the same, Δ *error* gives an insight of how the algorithm is degraded by scale and noise.

Experiment	Dataset	Algorithm	MQE	Δ error
1	*Equal Cond*	PCA	0.0157	-
		Proposed	**0.0087**	-
2	*Diff Light 1*	PCA	0.3011	19.17 (*Equal Cond*)
		Proposed	**0.0092**	**1.05** (*Equal Cond*)
	Diff Light 2	PCA	0.1244	7.892 (*Equal Cond*)
		Proposed	**0.0095**	**1.09** (*Equal Cond*)
	Diff Light 3	PCA	0.2903	18.49 (*Equal Cond*)
		Proposed	**0.0096**	**1.01** (*Equal Cond*)
3	*Different Dist 1*	PCA	0.1476	**1.19** (*Diff Light 2*)
		Proposed	**0.0311**	3.27 (*Diff Light 2*)
	Different Dist 2	PCA	0.3601	**1.24** (*Diff Light 3*)
		Proposed	**0.0579**	6.03 (*Diff Light 3*)
4	*Diff Light 2* + Blur Noise	PCA	0.1458	**1.17** (*Diff Light 2*)
		Proposed	**0.0156**	1.64 (*Diff Light 2*)
	Diff Light 3 + Gauss Noise	PCA	0.4267	**1.47** (*Diff Light 3*)
		Proposed	**0.0977**	10.18 (*Diff Light 3*)

Table 4.3: Summary with MQE and Δ *error* over the different experiments.

Algorithm	Computational time per image (seconds)
PCA	**0.0092**
Proposed	0.85

Table 4.4: Computational time of the PCA and proposed approach.

4.3.1 Computational Time

Regarding the computational time, the PCA based approach from [3] outperforms the proposed approach by a factor of 100 (see Table 4.4). This is due to two reasons: First, the PCA based algorithm avoids the computation of the HOG descriptor. Second, the elements of the principal components of PCA are scalars, whereas the used principal component algorithm (Sparse 2DPCA-L_1) produces principal components formed by vector elements, due to its two-dimensional nature. In combination, the proposed approach has a codebook with larger entries and an additional descriptor computation step. Performing a profiling of the code, the most time-consuming module is the codebook matching itself. Suggestions for improving the computational time are described in the future work (Section 5.2)

4.3.2 Invariance to Illumination

Experiment 2 shows that the proposed approach greatly outperforms the PCA based algorithm in a set of different illumination conditions by an order of magnitude in terms of MQE (see

Experiment 2 from Table 4.3). In addition, the proposed approach converges much faster to low $dist_{\%}$ values as Figures 4.5, 4.6, and 4.7 show. This suggests that the proposed solution generalizes much better to light configurations not present in the codebook, such as different overall lightings, shadows, and over-exposed areas. This is also reflected by the Δ *error*, where results of the PCA approach are one order of magnitude worse when the illumination conditions change. In comparison, the proposed approach yields similar performance (Δ *error* = 1.01 and 1.10). This is encouraged by the use of gradient-based descriptors that have been proven to be robust to different illumination conditions. In addition, in [25], sparse 2DPCA-L_1 is able to find principal components that generalize better to illumination changes.

4.3.3 Invariance to Scale

The invariance to scale in the proposed application is not a major issue as the distances considered do not vary greatly. In any case, both approaches generalized well to scale. However, PCA is more robust to scale changes, it has a Δ *error* of 1.19 and 1.24 for the sets of 75 and 100 meters respectively (see Experiment 3 from Table 4.3). Whereas the proposed approach yields MQE 3.27 and 6.09 times bigger than the reference dataset. This is also reflected in the convergence rates of the $dist_{\%}$ measure (see Figures 4.9 and 4.10), that are decreased when the distance increases. In our opinion, this is caused by the fact that HOG descriptors are not scale invariant, penalizing the performance when the distance between the camera and the target increases. For alternative applications, where larger distances have to be considered, a larger codebook parametrized by the distance may be used.

4.3.4 Distance Estimation

The proposed distance estimation stage using a monocular camera does not provide instant accurate measurements. This is caused by two main reasons: the same exact attitudes used to generate the codebook are rarely the same in the testing stage, hence, the observed height in test may differ from the one in the codebook. Besides, the process to extract the pixel-height of the spacecraft in the image is an approximate procedure, that may be affected by noise, as the number of pixels tagged as spacecraft may vary, and then the observed pixel height. However, the distance can be approximated if the instant measurements are averaged over time (see Figures 4.11 and 4.12).

4.3.5 Invariance to Noise

The results of Experiment 4 are similar as those in Experiment 3 (see Figures 4.14 and 4.15). Although the proposed approach achieves better classification rates, the PCA-based approach generalizes better for the two types of noise studied. The error increase factors of PCA are around 1.17 for the motion and 1.47 for the Gaussian Noise, compared to 1.64 and 10.18 for the proposed approach. It is interesting to see that the proposed approach is able to generalize well to motion blur, even though the HOG descriptor part may be affected by this noise. This may be encouraged by the use of intensity descriptors that are able to make up for the behaviour in HOG on blurred images. Respect to the Gaussian noise, the high-frequency patterns may affect the HOG descriptors, that are densely sampled on the spacecraft, resulting in very different representations from the training images used to build the codebook.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This master's thesis has presented a novel system for spacecraft pose estimation under non-cooperative rendezvous scenarios. The system is based on a feature codebook, where a subset of attitudes are described by means of intensity and HOG descriptors. To achieve an efficient representation of the descriptors, they are projected onto their principal components computed via the sparse 2DPCA-L_1 algorithm. The attitudes are represented by means of random quaternions that allow to obtain an uniformly sampled set of attitudes and unique representations of the rotations.

In addition, exploiting the *a priori* knowledge from the spacecraft 3D-model, a method for estimating the distance using a monocular camera has been proposed. The method provides good approximations of the distance if the instantaneous measurements are averaged over time.

The analysis of the experiments shows that the requirements of the algorithm (see Section 1.2.3) are met:

- Uncooperative target: The algorithm does not rely on any *a priori* information of the attitude, in fact, the experiments are designed in a way that the attitudes used for building the codebook are not likely to be present in the test datasets.
- Illumination Changes: The algorithm has been tested under three different illumination conditions, with varying intensities and incident angles. Experiments show that the proposed approach was able to achieve good classification rates on all the considered

scenarios. This is encouraged by the use of HOG descriptors and the application of sparse 2DPCA-L_1 on the pixel intensity values.

- High symmetry: The formulation of attitudes with quaternions and the use of holistic, rotation invariant descriptors allow to describe uniquely each possible view. This is advantageous respect to widely used local methods that model each view as a of descriptors that can be present in different attitudes. In addition, HOG and the pixel intensity values are not rotation invariant, hence, removing the ambiguity caused by two similar rotated views of the spacecraft.
- Real time processing requirement: The algorithm was able to classify images at $\sim$1Hz under an unoptimized MATLAB implementation on a desktop processor. Suggestions for improvement in this line are proposed in Section 5.2.

In addition, the algorithm was tested to measure the invariance to scale and noise. In the case of study presented, the proposed system shows enough invariance to scale and blur motion noise. Unfortunately, high presence of Gaussian noise affects the classification, which suggests that cameras with high SNR capture devices or additional de-noising stages may be considered.

5.2 Future Work

The presented master's thesis studies a novel codebook approach for spacecraft pose estimation. During the review of the state of the art, the only found approach based on codebook was the work developed by Shi and Ulrich in [3]. Therefore, there is room for improvement in this line of research, including the following ones that define the areas of future work:

- Efficient codebook matching: The computational cost of matching the codebook grows when the dimensionality of the features or the number of attitudes is increased. The suggested lines of research here are the study and implementation of faster approximate nearest-neighbour matchers, or efficient tree-based codebooks to reduce the search space.
- Scale invariant features: The use of HOG features allows to achieve good invariance to illumination changes, but HOG is not robust to scale. Alternative approaches to increase the robustness to scale such as pyramidal features may be studied.
- Study of different descriptors: The use of binary descriptors such as BRISK [33], densely sampled over the spacecraft may provide an efficient and robust feature to enhance the codebook representativeness.

Appendix A

Histogram of Oriented Gradients

Histogram of Oriented Gradients is a descriptor proposed by Dalal and Triggs in [30]. It encodes the structure of the object by means of describing how the gradients are oriented along the image. First, the image is divided into small spatial regions known as cells. In each cell, the orientations of the borders are described by means of one dimensional histogram, that accumulates the different present orientations.

Then several cells are merged to conform a block where the energy inside the block is computed and used to normalize the cells present in the histogram. In this way, better invariance to illumination changes is achieved. The set of normalized blocks conform the HOG descriptor.

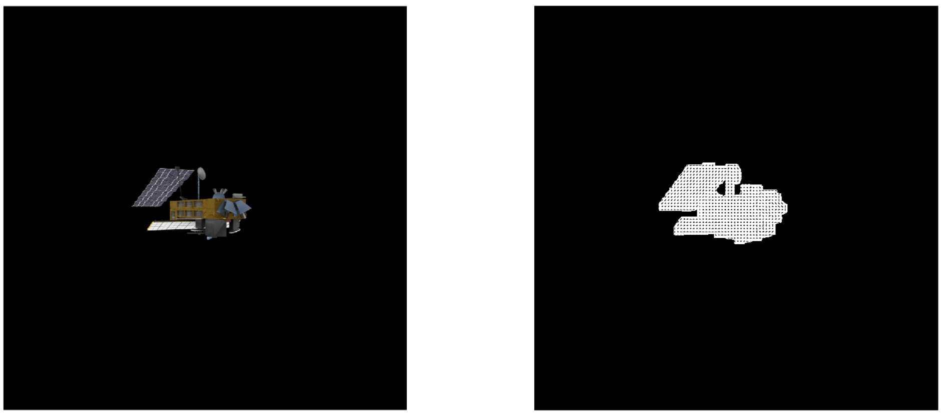

FIGURE A.1: Histogram of Oriented Gradients representation from an Envisat image.

Bibliography

[1] Yang Liu, Zongwu Xie, Bin Wang, and Hong Liu. Pose measurement of a non-cooperative spacecraft based on circular features. In *Real-time Computing and Robotics (RCAR), IEEE International Conference on*, pages 221–226. IEEE, 2016.

[2] Farhad Aghili, Marcin Kuryllo, Galina Okouneva, and Chad English. Robust vision-based pose estimation of moving objects for automated rendezvous & docking. In *Mechatronics and Automation (ICMA), 2010 International Conference on*, pages 305–311. IEEE, 2010.

[3] Jian-Feng Shi, Steve Ulrich, and Stephane Ruel. Spacecraft pose estimation using principal component analysis and a monocular camera. In *AIAA Guidance, Navigation, and Control Conference*, page 1034, 2017.

[4] James R. Wertz and Robert Bell. Autonomous rendezvous and docking technologies: Status and prospects. *Space Systems Technology and Operations*, Jun 2003. doi: 10.1117/12.498121.

[5] European Space Agency. Esa - e.deorbit. URL `http://www.esa.int/Our_Activities/Space_Engineering_Technology/Clean_Space/e.Deorbit`.

[6] J.A.F. Deloo. Analysis of the rendezvous phase of e.deorbit. guidance, communication and illumination. Master's thesis, Delft University of Technology, 2014.

[7] Thomas Schildknecht. Optical surveys for space debris. pages 46–54, January 2007. doi: 10.1007/s00159-006-0003-9.

[8] Clean Space. e.deorbit implementation plan, December 2015. URL `http://blogs.esa.int/cleanspace/files/2016/02/e.deorbit-Implementation-Plan-Issue_1_2015.pdf`.

[9] Jesus Gil and Guillermo Ortega. Esa developments on gnc systems for non-cooperative rendezvous, May 2016. URL `https://indico.esa.int/indico/event/128/material/5/6.pdf`.

[10] J.m. Kelsey, J. Byrne, M. Cosgrove, S. Seereeram, and R.k. Mehra. Vision-based relative pose estimation for autonomous rendezvous and docking. *2006 IEEE Aerospace Conference*. doi: 10.1109/aero.2006.1655916.

[11] Shai Segal, Avishy Carmi, and Pini Gurfil. Vision-based relative state estimation of non-cooperative spacecraft under modeling uncertainty. In *Aerospace Conference, 2011 IEEE*, pages 1–8. IEEE, 2011.

[12] Matthew D Lichter and Steven Dubowsky. Estimation of state, shape, and inertial parameters of space objects from sequences of range images. In *Proc. of SPIE Vol*, volume 5267, page 195, 2003.

[13] Xiaodong Du, Bin Liang, and Yanbo Tao. Pose determination of large non-cooperative satellite in close range using coordinated cameras. In *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, pages 3910–3915. IEEE, 2009.

[14] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.

[15] Sumant Sharma et al. Comparative assessment of techniques for initial pose estimation using monocular vision. *Acta Astronautica*, 123:435–445, 2016.

[16] Antoine Petit, Eric Marchand, and Keyvan Kanani. Vision-based space autonomous rendezvous: A case study. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 619–624. IEEE, 2011.

[17] Antoine Petit, Eric Marchand, and Keyvan 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.

[18] Lindsay I Smith et al. A tutorial on principal components analysis. *Cornell University, USA*, 51(52):65, 2002.

[19] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[20] Feiping Nie, Jianjun Yuan, and Heng Huang. Optimal mean robust principal component analysis. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1062–1070. PMLR, 2014.

[21] Jian Yang, David Zhang, Alejandro F Frangi, and Jing-yu Yang. Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 26(1):131–137, 2004.

[22] Hui Kong, Xuchun Li, Lei Wang, Earn Khwang Teoh, Jian-Gang Wang, and Ronda Venkateswarlu. Generalized 2d principal component analysis. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 1, pages 108–113. IEEE, 2005.

[23] Parinya Sanguansat. Two-dimensional principal component analysis and its extensions. In *Principal Component Analysis*. InTech, 2012.

[24] Xuelong Li, Yanwei Pang, and Yuan Yuan. L1-norm-based 2dpca. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(4):1170–1175, 2010.

[25] Haixian Wang and Jing Wang. 2dpca with l1-norm for simultaneously robust and sparse modelling. *Neural Networks*, 46:190–198, 2013.

[26] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.

[27] Ivan Selesnick. Introduction to sparsity in signal processing. *Connexions*, 2012.

[28] James J Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3993–3998. IEEE, 2004.

[29] Xavier Perez-Sala, Laura Igual, Sergio Escalera, and Cecilio Angulo. Uniform sampling of rotations for discrete and continuous learning of 2d shape models. In *Robotic Vision: Technologies for Machine Learning and Vision Applications*, pages 23–42. IGI Global, 2013.

[30] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[31] Grant Borodin. Envisat 3d model. URL `https://sketchfab.com/models/65b0ec49681a44f68dfc8bd4efe95839`.

[32] Blender 3d creation suite. URL `https://www.blender.org`.

[33] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.