



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS UNIVERSITÁRIO, CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Guilherme Henrique dos Santos

Static Attitude Determination Using Convolutional Neural Networks

Florianópolis
2021

Guilherme Henrique dos Santos

Static Attitude Determination Using Convolutional Neural Networks

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina para a obtenção do título de Mestre em Engenharia Elétrica.

Advisor: Prof. Eduardo Augusto Bezerra, PhD.

Florianópolis

2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Santos, Guilherme Henrique dos
Static Attitude Determination Using Convolutional
Neural Networks / Guilherme Henrique dos Santos ;
orientador, Eduardo Augusto Bezerra, 2021.
76 p.

Dissertação (mestrado) - Universidade Federal de Santa
Catarina, Centro Tecnológico, Programa de Pós-Graduação em
Engenharia Elétrica, Florianópolis, 2021.

Inclui referências.

1. Engenharia Elétrica. 2. Determinação de atitude. 3.
Redes neural. 4. Incerteza. I. Bezerra, Eduardo Augusto.
II. Universidade Federal de Santa Catarina. Programa de Pós
Graduação em Engenharia Elétrica. III. Título.

Guilherme Henrique dos Santos

Static Attitude Determination Using Convolutional Neural Networks

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Danilo Silva, Dr.
Universidade Federal de Santa Catarina

Prof. Laio Oriel Seman, Dr.
Universidade Federal de Santa Catarina

Prof. Luiz Fernando Carvalho, Dr.
Universidade Tecnológica Federal do Paraná

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Mestre em Engenharia Elétrica.

Coordenação do Programa de
Pós-Graduação

Prof. Eduardo Augusto Bezerra, PhD.
Supervisor

Florianópolis, 2021.

Este trabalho é dedicado aos meus pais, irmã e aos meus
amigos que me acompanharam e deram apoio durante o
processo.

AGRADECIMENTOS

Aos meus pais Jaimira e José e irmã Thayzze pelas oportunidades, ensinamentos, compreensão e ajuda em diversos momentos difíceis da minha vida. Aos meus amigos de infância Heitor, Icaro e Matheus Franco, que apesar da distância sempre me apoiaram. Ao meu primo Eduardo e amigos da graduação Áthur, Carla, Daniel Nesvera, Felipe Albeche, Francisca, Gabriel de Jesus, Guilherme Christmann, Lucas, Mariana, Ruan e Wilian Padilha que me trazem diferentes perspectivas sobre a vida. Ao meu orientador professor Eduardo Augusto Bezerra e professor Laio Oriel Seman pelas dicas, paciência e liberdade proporcionadas para seguir com o projeto.

Agradeço também à Universidade Federal de Santa Catarina junto com seu corpo docente e funcionários por proporcionarem um espaço tão agradável não somente para estudar, mas também para passear e descansar a mente em momentos de turbulência.

RESUMO

A necessidade de estimar a orientação entre sistemas referenciais é crucial na navegação espacial. Apesar dos algoritmos clássicos para este tipo de problema dependerem de abordagens mais algébricas, soluções baseadas em dados estão tornando-se mais atrativas devido a sua natureza estocástica. Conseqüentemente, uma abordagem baseada em redes neurais convolucionais para a determinação de atitude estática foi proposta. Diversos modelos foram treinados com distintos conjuntos de dados, estes contendo diferentes quantidades de vetores de observação. Tais vetores foram utilizados na construção da entrada do sistema, chamada matriz de perfil de atitude. Ao aumentar o número de vetores de observação, o modelo obteve melhores resultados sobre o conjunto de dados de validação, mas acabou não refletindo o verdadeiro comportamento do sistema quando apresentado a cenários desafiadores. Assim, a incerteza das previsões nesses cenários de teste foi levada em consideração na escolha do melhor modelo. Apesar do modelo não ter obtido melhores resultados em comparação com algoritmos tradicionais como SVD, q-Method, QUEST e ESOQ2, o modelo escolhido se mostrou menos sensível a ruídos mais elevados do que eles.

Palavras-chave: Determinação de atitude. Incerteza. Rede neural.

RESUMO EXTENDIDO

Introdução

Quando um satélite está seguindo um destino, ele precisa saber se está navegando pelo caminho correto. Para validar sua trajetória, ele deve possuir algum conhecimento a respeito de sua orientação baseando-se em um sistema de coordenadas de referência. Para obter essa informação, deve-se encontrar a diferença entre duas coleções de vetores. O primeiro conjunto, representado no sistema de coordenadas do corpo, é um conjunto de vetores obtidos através de sensores embutidos no satélite. Já o segundo, é uma representação destes dados mensurados, porém representados no sistema de coordenadas de referência. A diferença entre esses dois conjuntos é representada por uma operação de rotação, pois trata-se de um problema de rotação de um sistema definido por $(\vec{a}, \vec{b}, \vec{c})$ para outro definido por $(\vec{x}, \vec{y}, \vec{z})$. Vários algoritmos foram desenvolvidos para executar esta tarefa, alguns consideram estimativas anteriores para gerar novas rotações, chamados de determinação de atitude dinâmica. Outros consideram apenas medidas obtidas simultaneamente, chamados de determinação de atitude estática. Este último sendo o foco desta dissertação.

Objetivos

Os algoritmos usados na determinação de atitude estática consideram que suas entradas sejam obtidas simultaneamente. Desta forma, conseguem determinar a matriz de orientação ótima para aquele dado instante. Por outro lado, eliminar a variável tempo faz com que essa classe de algoritmos dependa da qualidade dos dados de entrada, ou seja, a precisão dos sensores afeta drasticamente seu desempenho. Logo, o objetivo desta dissertação é aplicar técnicas de aprendizado de máquina para atenuar o efeito do ruído de entrada nas predições do sistema. Será considerado um modelo baseado em redes neurais convolucionais.

Metodologia

Foi utilizada uma arquitetura conhecida como *PointNet*. Pois esta, além de aceitar nuvens de pontos como entrada, apresenta robustez em relação a ruídos de entrada. Foram feitas mudanças nesta arquitetura para adequá-la ao novo formato de entrada proposto. Nesta nova entrada, as nuvens de pontos foram utilizados para construir uma matriz, chamada matriz de perfil de atitude, esta que encapsula a similaridade entre duas coleções de vetores representados em sistemas de coordenadas distintos. Uma noção da incerteza do modelo foi estimada através da matriz de covariância das predições, com o auxílio de camadas *Dropout*. Em relação à base de dados, vetores referenciados no sistema de coordenadas de referência \vec{r} e matrizes de rotação \mathbf{R} foram gerados aleatoriamente, sendo os vetores referenciados no sistema de coordenadas do corpo \vec{b} o resultado da rotação de \vec{r} pela matriz \mathbf{R} . O modelo foi treinado considerando diferentes quantidades de vetores de observação e probabilidades de *dropout*, utilizando uma função de custo que avalia a diferença angular entre a matriz de rotação original e a predita.

Resultados e Discussão

Foram treinadas quatro arquiteturas, uma delas insere as coleções de vetores diretamente no modelo, enquanto que as demais utilizam a matriz de perfil de atitude \mathbf{B} como entrada. Foram testados dois tipos de funções de ativação: *Rectified Linear Unit* e *Swish*. Foi testado também a construção da matriz \mathbf{B} considerando dois cenários:

utilizando os pesos originais e utilizando os pesos iguais. Notou-se que utilizar **B** invés dos vetores originais resultou na melhoria significativa do desempenho da rede. Ao utilizar os pesos iguais, o modelo obteve menores valores de erro na função de Wahba tanto no treino quanto na validação se comparado com os pesos reais. A função *Swish* levou à melhores resultados em comparação a *Rectified Linear Unit*. O modelo se comportou melhor no banco de dados de validação ao aumentar o número de vetores de observação utilizados, porém esse comportamento não se manteve ao ser testado nos doze casos de teste considerados. Portanto, foram calculadas as incertezas desses modelos para cada teste a fim de determinar qual se sairia melhor. Essas incertezas foram calculadas com o auxílio da matriz de covariância, sendo assim possível obter a variância das predições em cada um dos eixos: **x**, **y** e **z**. Ao comparar o modelo escolhido com os algoritmos SVD, q-Method, QUEST e ESOQ2, foi possível perceber que o mesmo apresentou maior robustez quando os dados de entrada estavam sujeitos a maiores ruídos.

Considerações finais

Por mais que a rede não tenha se saído melhor que os algoritmos tradicionais, ela não sofreu perturbações significativas na qualidade de suas predições se comparado aos demais, mesmo nos casos de teste com maior ruído. Logo, ela é menos sensível a ruídos. Como a natureza dos dados segue uma estatística circular, seria melhor incorporar conceitos da distribuição de Bingham. Desta forma, poderia ser possível reduzir a incerteza e o erro da função de Wahba.

Palavras-chave: Determinação de atitude. Incerteza. Rede neural.

ABSTRACT

The need for estimating the orientation between frames of reference is crucial in spacecraft navigation. Although the classic algorithms for this problem rely on more algebraic approaches, data-driven solutions are becoming more appealing due to their stochastic nature. Hence, a method based on convolutional neural networks for the static attitude determination problem was proposed. Several models were trained using different datasets containing a different number of observation vectors. Such vectors were used to build the input to the system, called the attitude profile matrix. When increasing the number of observation vectors, the model obtained better results over the validation dataset. Still, it turned out not to reflect the system's actual behavior when facing challenging scenarios. Thus, the uncertainty of predictions on these test scenarios was considered when choosing the best model. The model did not manage to achieve as good results as traditional algorithms, such as SVD, q-Method, QUEST, and ESOQ2. However, the proposed model proved to be less sensitive to higher noise than the traditional algorithms.

Keywords: Attitude determination. Uncertainty. Neural network.

LIST OF FIGURES

Figure 1 – Compass as an attitude representation.	26
Figure 2 – Rotation from one frame to another.	26
Figure 3 – Coordinate frames.	29
Figure 4 – Euler axis/angle rotation.	31
Figure 5 – All rotations are executed from left to right where a) represents a XYZ rotation and b) represents a ZYX rotation.	35
Figure 6 – Representation mapping.	44
Figure 7 – Modified PointNet.	48
Figure 8 – Results over the validation dataset.	51
Figure 9 – The relationship of the p-values among all models.	54
Figure 10 – Critical difference diagram for Nemenyi tests. Bold lines indicate groups of models which are not significantly different (their average ranks differ by less than the $CD = 1.761$).	54
Figure 11 – Axes uncertainties for each test case considering a dropout rate of 10%.	55
Figure 12 – Axes uncertainties for each test case considering a dropout rate of 15%.	56
Figure 13 – Axes uncertainties for each test case considering a dropout rate of 20%.	56
Figure 14 – Dispersion of the absolute angular difference when using a dropout rate of 10%.	57
Figure 15 – Dispersion of the absolute angular difference when using a dropout rate of 15%.	58
Figure 16 – Dispersion of the absolute angular difference when using a dropout rate of 20%.	58
Figure 17 – Wahba’s error evolution for Test Case 8.	59
Figure 18 – Variance evolution for Test Case 8.	59
Figure 19 – Comparison of Wahba’s error evaluation among some algorithms.	60
Figure 20 – Impact of average measurement noise over model performance.	61
Figure 21 – A sample from the ShapeNet airplane category rotated by three different rotation matrices.	71
Figure 22 – Mean angular error during training and validation.	72
Figure 23 – Overall performance for three different ϕ_{max}.	73

Figure 24 – Evolution of the angular error when varying the number of consumed points.	73
Figure 25 – Model architectures, where a) was built considering the mapping function $6D \rightarrow SO(3)$, and b) considering the eigenvector extraction from a symmetric matrix.	74

LIST OF TABLES

Table 1 – Accuracy of common attitude sensors.	27
Table 2 – Alternative Representations for Three-Axis Attitude.	30
Table 3 – Wahba’s error for each model.	53

LIST OF ABBREVIATIONS AND ACRONYMS

ANN	<i>Artificial Neural Network</i>
AWGN	<i>Additive White Gaussian Noise</i>
CD	<i>Critical Difference</i>
CNN	<i>Convolution Neural Network</i>
DCM	<i>Direction Cosine Matrix</i>
ESOQ2	<i>Second Estimator of the Optimal Quaternion</i>
GPS	<i>Global Positioning System</i>
Mask R-CNN	<i>Mask Region Based Convolutional Neural Networks</i>
MCD	<i>Monte Carlo Dropout</i>
NaN	<i>Not a Number</i>
QUEST	<i>QUaternion ESTimator</i>
ReLU	<i>Rectified Linear Unit</i>
RNN	<i>Recurrent Neural Network</i>
SO(3)	<i>Special Orthogonal Group</i>
SVD	<i>Singular Value Decomposition</i>

LIST OF SYMBOLS

ϕ	Rotational angle about \vec{e} / Roll angle
\mathbf{A}	Orthogonal attitude matrix that minimizes Wahba's problem
\vec{e}	Unit vector that represents the rotational axis
ψ	Yaw angle
θ	Pitch angle
\vec{r}_i	Unit vector of an object of interest in the reference frame
\vec{b}_i	Unit vector of an object of interest in the spacecraft body frame
a_i	Weight that states how accurate is a measurement vector \vec{b}_i
λ_0	Sum of all weights of each pair of observation vectors
\mathbf{B}	Attitude profile matrix
\mathbf{K}	Davenport's traceless matrix
\vec{q}_{opt}	Optimal rotation represented as a unit quaternion
\mathbf{R}	Groundtruth rotation matrix
$\hat{\mathbf{R}}$	Predicted rotation matrix

CONTENTS

1	Introduction	25
1.1	OBJECTIVE	28
2	Background	29
2.1	ATTITUDE MATRIX CONSTRUCION	29
2.2	ATTITUDE REPRESENTATIONS	31
2.2.1	Euler Axis/Angle	31
2.2.2	Quaternions	32
2.2.3	Gibbs-Rodrigues Vector	33
2.2.4	Euler Angles	34
2.3	THREE-AXIS ATTITUDE DETERMINATION	36
2.3.1	Wahba's Problem	36
2.3.2	Solutions for Wahba's Problem	37
2.3.2.1	Davenport's q-Method	37
2.3.2.2	SVD based	38
2.3.2.3	QUEST	39
2.3.2.4	ESOQ2	40
3	Related Work	43
4	Experiments	47
4.1	DATASET	47
4.2	NEURAL NETWORK ARCHITECTURE	47
4.3	TRAINING	49
4.4	MODEL CHOICE	51
4.5	COMPARISON WITH TRADITIONAL ALGORITHMS	60
5	Conclusion	63
	REFERENCES	65
	APPENDIX A EVALUATION ON SHAPENET	71

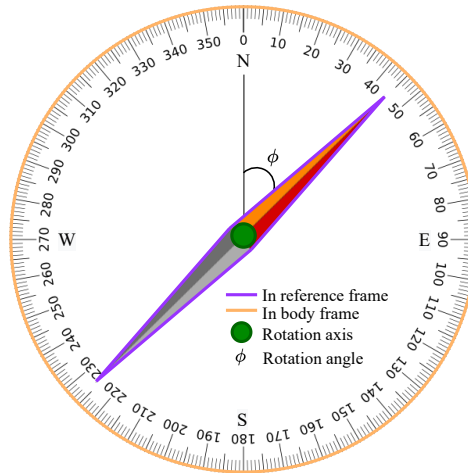
1 INTRODUCTION

Spacecrafts have targets, and they need to know whether they are on the correct path. To validate the trail, a spacecraft must have some knowledge about its orientation based on a reference frame. For example, considering a boat targeting an island located in the same direction of Earth's north magnetic pole. Considering also that the only available information about the boat's position is through a compass. After a while, the boat changes its trajectory, and the compass arrow points to the northeast. From the boat's point of view (body frame), one could say that the island has moved slightly to the east. However, the island cannot move as it is fixed concerning the Earth (reference frame). So, how should the boat be moved to point back to the north? The compass holds this information.

The compass is sensible to Earth's north magnetic pole since its needle always points towards there. Moreover, the compass can capture the boat's orientation as it is inside the vessel, then any change in the boat's direction forces the arrow to rotate. Hence, the compass can translate the orientation difference between the vessel and the Earth's magnetic north. The angular difference given by the compass tells us how we should move the boat to point it in the right direction.

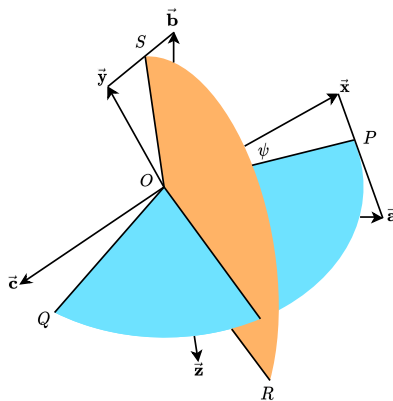
In this scenario, there are two objects in distinct reference frames: The boat and the island. The former is defined in the body frame, and the latter in the reference frame. The compass acts as an attitude representation since it provides information about the difference between the two frames and how one should change the boat's direction. The rotation axis, which dictates the spinning direction, and the rotation angle, which says how much this boat should be turned, parameterize this orientation difference. In Figure 1, the green circle in the middle, which points out of the paper, represents the axis of rotation, and ϕ stands for the angular difference.

The above example is analogous to attitude determination since it provides information about the deviation between some objects of interest. The body frame and the desired frame of reference provide different perspectives about those objects. Thus, a rotation operation should determine their difference. To simplify, the goal is to find the rotation axis and the rotation angle that best rotates a frame defined by $(\vec{a}, \vec{b}, \vec{c})$ to another frame $(\vec{x}, \vec{y}, \vec{z})$. Finding rotation between frames is crucial in many fields, such as robotics, navigation and control, computer graphics, among others.

Figure 1: **Compass as an attitude representation.**

Source: Own authorship

Figure 2 displays that at least two vectors define a proper rotation between frames. The intersection between the planes OSR and OPQ defines the unique rotational axis that can align the two coordinate systems. In spacecraft navigation, the satellite's onboard sensors such as Earth sensors, Sun sensors, star sensors, *Global Positioning System* (GPS), and magnetometers return their acquired information as three dimensional vectors. Except for the star sensors, all the other sensors can estimate the attitude only in one direction, that is, the sensor obtains only one measurement vector. Hence, multiple sensors have to be used to get a valid three-axis attitude information. On the other hand, star sensors can return multiple vectors from a single measurement, leading to better accuracies while using fewer resources. The accuracy of attitude measurements of several attitude sensors can be seen in Table 1.

Figure 2: **Rotation from one frame to another.**

Source: Adapted from (YANG, 2019, p. 29)

Table 1: **Accuracy of common attitude sensors.**

	Reference frame	Attitude measurement accuracy
Earth sensor	Horizon	6 arcminutes
Sun sensor	Sun	1 arcminute
Magnetometer	Geomagnetism	30 arcminutes
Star sensor	Star	1 arcsecond

Source: (ZHANG, 2019)

Several algorithms have been developed throughout the years trying to determine the attitude based on pairs of vectors. These approaches can make use of stochastic analysis or not. The former, also known as attitude estimation or dynamic attitude determination, uses Kalman filters or maximum likelihood approaches, and requires some previous attitude information to adjust its predictions. Thus, it depends on the time variable. The latter, also known as static attitude determination, which is this dissertation’s focus, considers only the measurements taken simultaneously or close enough in time so that spacecraft motion between the observations can be ignored. It has as a requirement that there are enough measurements available for the attitude determination at each computation step.

In the computer vision field, there is a similar problem to the one described before, whose premise is to find the orientation of objects with respect to the camera. This task also shares the same goal as attitude determination, that is, it determines the orientation or pose of objects of interest from one coordinate frame to another. In the compute vision context, this task is referred to as pose estimation. Nowadays, researchers prefer to address this kind of problem by using machine learning techniques because these approaches can capture vast amounts of information from a single input signal without requiring manual feature extraction.

The pose estimation task considers three main scenarios: human pose estimation, 6D pose estimation, and 3D point cloud estimation. The first scenario focuses in determining the orientation of the human joints or contours based on images (BULAT et al., 2020; LIU et al., 2021) or videos (PAVLLO et al., 2019). The second one tries to estimate the 6D pose of objects contained in images along with their semantic masks and bounding boxes (BUKCHAT; VETTER, 2020; ZAKHAROV et al., 2019; IWASE et al., 2021). And last but not least, the 3D point cloud registration (HORACHE et al., 2021; CHOY et al., 2020; POIESI; BOSCAINI, 2021) focuses in determining the orientation representation that best aligns two sets of point clouds.

The focus of this work, as stated before, will be the application of machine learning techniques for three-axis attitude determination, given that data-driven approaches have the ability to leverage interesting insights about real-world problems. Furthermore, such methods can take the intrinsic randomness of the task in consideration while training since it follows a stochastic procedure. Therefore, it can potentially mitigate the impact of noise on the input data. For static attitude determination, the attention on the variability of

the predictions may offer a promising advantage over the traditional algorithms because it can improve the quality of the predictions for noisy input data.

This dissertation is organized in chapters as follows: The second one presents a review of how rotation matrices can be constructed. It was divided in sections, where each one describes different options for attitude representations, demonstrating their pros and cons and their typical applications. Besides that, the second chapter discusses some of the widely used traditional algorithms. Moreover, this section explains how to develop such algorithms and provides some points that should be considered while elaborating them.

The third chapter discusses the related work, presenting a summary of the works that served as inspiration for the proposed model, pointing their weaknesses and strengths. The fourth chapter details the procedures considered in the model development. It describes how the dataset is created, the training plan of action along with the considered parameters, a method for choosing the best trained neural network model considering their level of uncertainty in the \mathbf{x} , \mathbf{y} and \mathbf{z} axes. It also presents a comparison between the chosen model and the traditional algorithms over Wahba's loss function.

Finally, the conclusion chapter exposes some reflections about the results obtained in the experiment chapter. Moreover, the appendix chapter carries out some tests involving the well-known ShapeNet dataset, which demonstrates the performance of the model presented in this dissertation against two recent works which tackle the same problem.

1.1 OBJECTIVE

The algorithms that solve static attitude determination consider that their inputs are observation vectors taken simultaneously or close enough in time to ignore spacecraft motion between the measurements. Therefore, they can determine the attitude at that given instant. Conversely, eliminating the time variable makes this class of algorithms susceptible to the quality of the input data, that is, the precision of the sensors drastically affects the integrity of the results. Due to their nature, they have a crucial role in the *lost-in-space* scenario (SPRATLING; MORTARI, 2009), where the spacecraft has no previous information about its attitude. Therefore, the navigation system must instantaneously infer an orientation relying on the quality of measurements taken by its sensors since it directly affects the performance of the algorithms (HASHIM, 2020).

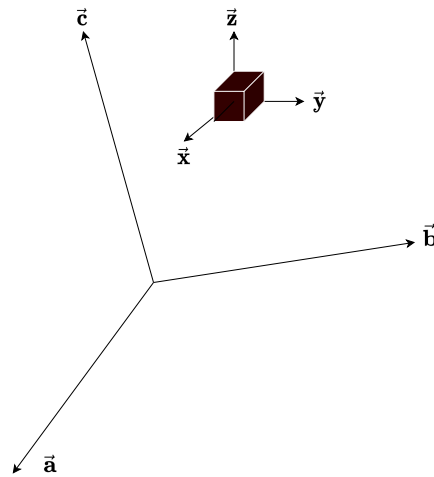
This Master Thesis targets the use of machine learning techniques to mitigate the impact of the measurements' perturbations over the predictions. Specifically, this work proposes a neural network-based approach for determining the rotation difference between two coordinate frames based on the *Convolution Neural Network* (CNN) architecture. We considered the model uncertainty in our strategy for choosing the model. Furthermore, we will demonstrate a performance comparison between the proposed model and classical algorithms.

2 BACKGROUND

2.1 ATTITUDE MATRIX CONSTRUCTION

Let us suppose that the position of a rigid body in space is represented in a coordinate system (body frame) where its basis vectors are \vec{x} , \vec{y} , and \vec{z} , such that they are orthonormal to each other. And, we want to know how to represent this position in another orthonormal coordinate frame (reference frame), where its basis vectors are \vec{a} , \vec{b} , and \vec{c} , as shown in Figure 3.

Figure 3: **Coordinate frames.**



Source: Own authorship

This can be achieved by using a rotation matrix \mathbf{A} , also called *attitude matrix*:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} \quad (1)$$

Each element of \mathbf{A} represents the cosine of the angle between the body unit vector and a reference axis, for this reason this matrix is also called *Direction Cosine Matrix* (DCM). "The direction cosine matrix is a coordinate transformation that maps vectors from the reference frame to the body frame" (WERTZ, 1978, p. 411), that is, to bring a vector \vec{k} with coordinates $(\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3)$ in the reference frame to the rigid body coordinate frame a rotation must be applied to this vector, as follows:

$$\vec{v} = \mathbf{A} \vec{k} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{k}_1 \\ \mathbf{k}_2 \\ \mathbf{k}_3 \end{bmatrix} \quad (2)$$

Table 2: Alternative Representations for Three-Axis Attitude.

PARAMETRIZATION	NOTATION	ADVANTAGES	DISADVANTAGES	COMMON APPLICATIONS
Direction Cosine Matrix	$\mathbf{A} = [A_{ij}]$	No singularities; No trigonometric functions; Convenient product rule for successive rotations;	Six redundant parameters;	In analysis, to transform from one reference frame to another;
Euler Axis/Angle	\vec{e}, ϕ	Clear physical interpretation;	One redundant parameter; Axis undefined when $\sin \phi = 0$; Trigonometric functions;	Commanding slew maneuvers;
Quaternions	$\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4$	No trigonometric functions; Convenient product rule for successive rotations;	One redundant parameter; No obvious physical interpretation; Discontinuous at a 180° rotation angle;	Onboard inertial navigation;
Gibbs vector	\vec{g}	No redundant parameters; No trigonometric functions; Convenient product rule for successive rotations;	Infinite for 180° rotation;	Analytic studies;
Euler angles	ϕ, θ, ψ	No redundant parameters; Physical interpretation is clear in some cases;	Trigonometric functions; Singularity at some θ ; No convenient product rule for successive rotations;	Analytic studies; Input/Output; Onboard attitude control of 3-Axis stabilized spacecraft;

Source: Adapted from (WERTZ, 1978, p. 412)

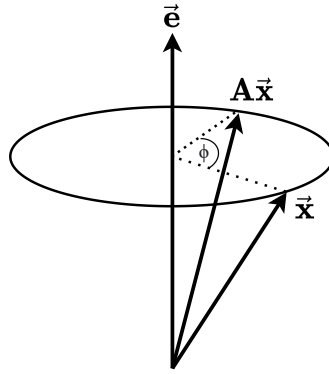
Since \mathbf{A} represents a rotation, it follows that it is an orthogonal matrix, implying that $\mathbf{A}^{-1} = \mathbf{A}^T$, therefore to map vectors from body frame back to reference frame, one just needs to do $\mathbf{A}^T \vec{\mathbf{v}} = \vec{\mathbf{k}}$. However, DCM matrices are not the only way to express rotations. Other forms can be seen in Table 2, where each one has its pros and cons.

2.2 ATTITUDE REPRESENTATIONS

2.2.1 Euler Axis/Angle

Since the attitude matrix \mathbf{A} is a proper real orthogonal 3x3 matrix, it follows that it has at least one eigenvector $\vec{\mathbf{e}}$ with an eigenvalue equal to one. Based on this property, we know that the unit vector $\vec{\mathbf{e}}$ has the same components both in the reference and body frame. Therefore, this vector lies along the axis of rotation. To effectively perform a rotation, we need one more piece of information: the rotation angle ϕ about the rotation axis. The parameters $\vec{\mathbf{e}}$ and ϕ are known as *Euler axis* and *Euler angle*, respectively.

Figure 4: **Euler axis/angle rotation.**



Source: Adapted from (MARKLEY; CRASSIDIS, 2014, p. 41)

The rotation of a vector $\vec{\mathbf{x}}$ through an angle ϕ about a rotation axis $\vec{\mathbf{e}}$ is depicted in Figure 4, where \mathbf{A} is a DCM parametrized by these two components. This matrix is defined as follows:

$$\mathbf{A} = \begin{bmatrix} \cos\phi + \mathbf{e}_1^2(1 - \cos\phi) & \mathbf{e}_1\mathbf{e}_2(1 - \cos\phi) + \mathbf{e}_3\sin\phi & \mathbf{e}_1\mathbf{e}_3(1 - \cos\phi) - \mathbf{e}_2\sin\phi \\ \mathbf{e}_1\mathbf{e}_2(1 - \cos\phi) - \mathbf{e}_3\sin\phi & \cos\phi + \mathbf{e}_2^2(1 - \cos\phi) & \mathbf{e}_2\mathbf{e}_3(1 - \cos\phi) + \mathbf{e}_1\sin\phi \\ \mathbf{e}_1\mathbf{e}_3(1 - \cos\phi) + \mathbf{e}_2\sin\phi & \mathbf{e}_2\mathbf{e}_3(1 - \cos\phi) - \mathbf{e}_1\sin\phi & \cos\phi + \mathbf{e}_3^2(1 - \cos\phi) \end{bmatrix} \quad (3)$$

If an attitude matrix is given, then we can get the corresponding axis and angle

back by using the following equations:

$$\phi = \arccos\left(\frac{\text{tr}(\mathbf{A}) - 1}{2}\right) \quad (4a)$$

$$\vec{\mathbf{e}} = \frac{1}{2\sin\phi} \begin{bmatrix} \mathbf{A}_{23} - \mathbf{A}_{32} \\ \mathbf{A}_{31} - \mathbf{A}_{13} \\ \mathbf{A}_{12} - \mathbf{A}_{21} \end{bmatrix} \quad (4b)$$

2.2.2 Quaternions

This type of parametrization, also known as *Euler symmetric parameters* or *Euler-Rodrigues parameters*, is widely used in many applications, such as computer graphics, computer vision, robotics, navigation, molecular dynamics, etc. The following components represent the quaternions:

$$\vec{\mathbf{q}} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \mathbf{q}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 \sin \frac{\phi}{2} \\ \mathbf{e}_2 \sin \frac{\phi}{2} \\ \mathbf{e}_3 \sin \frac{\phi}{2} \\ \cos \frac{\phi}{2} \end{bmatrix} \quad (5)$$

where \mathbf{e}_{1-3} are the components of a unit vector along the axis of rotation, and ϕ is the rotation angle about this axis. "The beauty of the quaternion representation is that it expresses the attitude matrix as a homogenous quadratic function of the elements of the quaternion, requiring no trigonometric or other transcendental function evaluations." (MARKLEY; CRASSIDIS, 2014, p. 46). This representation has no singularities and is more efficient because it has only four parameters instead of nine, leading to reduced memory consumption and computation time.

Quaternions that represent rotations must follow a single constraint: they must have unit norm, this implies that $\mathbf{q}_1^2 + \mathbf{q}_2^2 + \mathbf{q}_3^2 + \mathbf{q}_4^2 = 1$. As exemplified in Table 2, quaternions (unit quaternions to be more specific) have the disadvantage of not being physically interpretable because their domain space is embedded in a four-dimensional space. Despite having only four parameters, one is redundant because we are trying to represent three dimensions using a four-dimensional representation. A *direction cosine matrix* also can be built from quaternion representation as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{q}_1^2 - \mathbf{q}_2^2 - \mathbf{q}_3^2 + \mathbf{q}_4^2 & 2(\mathbf{q}_1\mathbf{q}_2 + \mathbf{q}_3\mathbf{q}_4) & 2(\mathbf{q}_1\mathbf{q}_3 - \mathbf{q}_2\mathbf{q}_4) \\ 2(\mathbf{q}_1\mathbf{q}_2 - \mathbf{q}_3\mathbf{q}_4) & -\mathbf{q}_1^2 + \mathbf{q}_2^2 - \mathbf{q}_3^2 + \mathbf{q}_4^2 & 2(\mathbf{q}_2\mathbf{q}_3 + \mathbf{q}_1\mathbf{q}_4) \\ 2(\mathbf{q}_1\mathbf{q}_3 + \mathbf{q}_2\mathbf{q}_4) & 2(\mathbf{q}_2\mathbf{q}_3 - \mathbf{q}_1\mathbf{q}_4) & -\mathbf{q}_1^2 - \mathbf{q}_2^2 + \mathbf{q}_3^2 + \mathbf{q}_4^2 \end{bmatrix} \quad (6)$$

If an attitude matrix is given, then we can get the corresponding quaternion by choosing one of the following vectors according to (MARKLEY, 2008):

$$\vec{\mathbf{x}}^{(1)} = 4\mathbf{q}_1\mathbf{q} = \begin{bmatrix} 1 + 2\mathbf{A}_{11} - \text{tr}(\mathbf{A}) \\ \mathbf{A}_{12} + \mathbf{A}_{21} \\ \mathbf{A}_{13} + \mathbf{A}_{31} \\ \mathbf{A}_{23} - \mathbf{A}_{32} \end{bmatrix} \quad (7a)$$

$$\vec{\mathbf{x}}^{(2)} = 4\mathbf{q}_2\mathbf{q} = \begin{bmatrix} \mathbf{A}_{21} + \mathbf{A}_{12} \\ 1 + 2\mathbf{A}_{22} - \text{tr}(\mathbf{A}) \\ \mathbf{A}_{23} + \mathbf{A}_{32} \\ \mathbf{A}_{31} - \mathbf{A}_{13} \end{bmatrix} \quad (7b)$$

$$\vec{\mathbf{x}}^{(3)} = 4\mathbf{q}_3\mathbf{q} = \begin{bmatrix} \mathbf{A}_{31} + \mathbf{A}_{13} \\ \mathbf{A}_{32} + \mathbf{A}_{23} \\ 1 + 2\mathbf{A}_{33} - \text{tr}(\mathbf{A}) \\ \mathbf{A}_{12} - \mathbf{A}_{21} \end{bmatrix} \quad (7c)$$

$$\vec{\mathbf{x}}^{(4)} = 4\mathbf{q}_4\mathbf{q} = \begin{bmatrix} \mathbf{A}_{23} - \mathbf{A}_{32} \\ \mathbf{A}_{31} - \mathbf{A}_{13} \\ \mathbf{A}_{12} - \mathbf{A}_{21} \\ 1 + \text{tr}(\mathbf{A}) \end{bmatrix} \quad (7d)$$

Since each of the $\vec{\mathbf{x}}^{(i)}$ is a scalar multiple of $\vec{\mathbf{q}}$, we can obtain our unit quaternion by normalizing any of the vectors in Equation (7). Although, we should choose the $\vec{\mathbf{x}}^{(i)}$ corresponding to the maximum value of \mathbf{q}_i^2 to minimize numerical errors.

$$\vec{\mathbf{q}} = \pm \frac{\vec{\mathbf{x}}^{(i)}}{\|\vec{\mathbf{x}}^{(i)}\|} \quad (8)$$

2.2.3 Gibbs-Rodrigues Vector

This parametrization, based on the *Rodrigues parameters*, was first introduced by Olinde Rodrigues in (RODRIGUES, 1840) and later represented by Josiah W. Gibbs in (GIBBS; WILSON, 1901) as a vector as we know nowadays. The vector denoted by $\vec{\mathbf{g}}$ is expressed as follows:

$$\mathbf{g}_1 = \frac{\mathbf{q}_1}{\mathbf{q}_4} = \mathbf{e}_1 \tan \frac{\phi}{2} \quad (9a)$$

$$\mathbf{g}_2 = \frac{\mathbf{q}_2}{\mathbf{q}_4} = \mathbf{e}_2 \tan \frac{\phi}{2} \quad (9b)$$

$$\mathbf{g}_3 = \frac{\mathbf{q}_3}{\mathbf{q}_4} = \mathbf{e}_3 \tan \frac{\phi}{2} \quad (9c)$$

where \mathbf{q}_{1-4} are the *Rodrigues parameters*, the same quaternion coefficients, and \mathbf{e}_{1-3} are the components of a unit vector along the axis of rotation. This vector "[...]" provides a

parametrization of three-dimensional rotations based on just three independent parameters [...] (VALDENEBRO, 2016), while quaternions depend on four parameters. A *direction cosine matrix* can be built from a Gibbs-Rodrigues representation as follows:

$$\mathbf{A} = \begin{bmatrix} 1 + \mathbf{g}_1^2 - \mathbf{g}_2^2 - \mathbf{g}_3^2 & 2(\mathbf{g}_1\mathbf{g}_2 + \mathbf{g}_3) & 2(\mathbf{g}_1\mathbf{g}_3 - \mathbf{g}_2) \\ 2(\mathbf{g}_1\mathbf{g}_2 - \mathbf{g}_3) & 1 - \mathbf{g}_1^2 + \mathbf{g}_2^2 - \mathbf{g}_3^2 & 2(\mathbf{g}_2\mathbf{g}_3 + \mathbf{g}_1) \\ 2(\mathbf{g}_1\mathbf{g}_3 + \mathbf{g}_2) & 2(\mathbf{g}_2\mathbf{g}_3 - \mathbf{g}_1) & 1 - \mathbf{g}_1^2 - \mathbf{g}_2^2 + \mathbf{g}_3^2 \end{bmatrix} \quad (10)$$

We can express this vector in terms of a DCM by using the set of Equations (11). This parametrization is not widely used, even providing a 1:1 mapping of rotations, because it becomes infinite for a 180° rotation angle.

$$\mathbf{g}_1 = \frac{1}{1 + \text{tr}(\mathbf{A})}(\mathbf{A}_{23} - \mathbf{A}_{32}) \quad (11a)$$

$$\mathbf{g}_2 = \frac{1}{1 + \text{tr}(\mathbf{A})}(\mathbf{A}_{31} - \mathbf{A}_{13}) \quad (11b)$$

$$\mathbf{g}_3 = \frac{1}{1 + \text{tr}(\mathbf{A})}(\mathbf{A}_{12} - \mathbf{A}_{21}) \quad (11c)$$

2.2.4 Euler Angles

We can express a rotation in terms of three rotation angles, known as *Euler angles*. To be more specific, we can break a single rotation into a sequence of three rotations using two intermediate frames throughout the process. Starting from an initial frame \mathcal{B} to a final frame \mathcal{R} , the resultant rotation $\mathbf{A}_{\mathcal{R}\mathcal{B}}$ is equivalent to perform a rotation from \mathcal{B} to a second frame \mathcal{I} , then from \mathcal{I} to \mathcal{K} , and finally from \mathcal{K} to \mathcal{R} . In order to do that, we just need to multiply each rotation matrix as follows:

$$\mathbf{A}_{\mathcal{R}\mathcal{B}} = \mathbf{A}_{\mathcal{R}\mathcal{K}}\mathbf{A}_{\mathcal{K}\mathcal{I}}\mathbf{A}_{\mathcal{I}\mathcal{B}} \quad (12)$$

where each matrix has a rotation angle and a rotation axis associated. Each rotation axis is a constant column vector selected from the set:

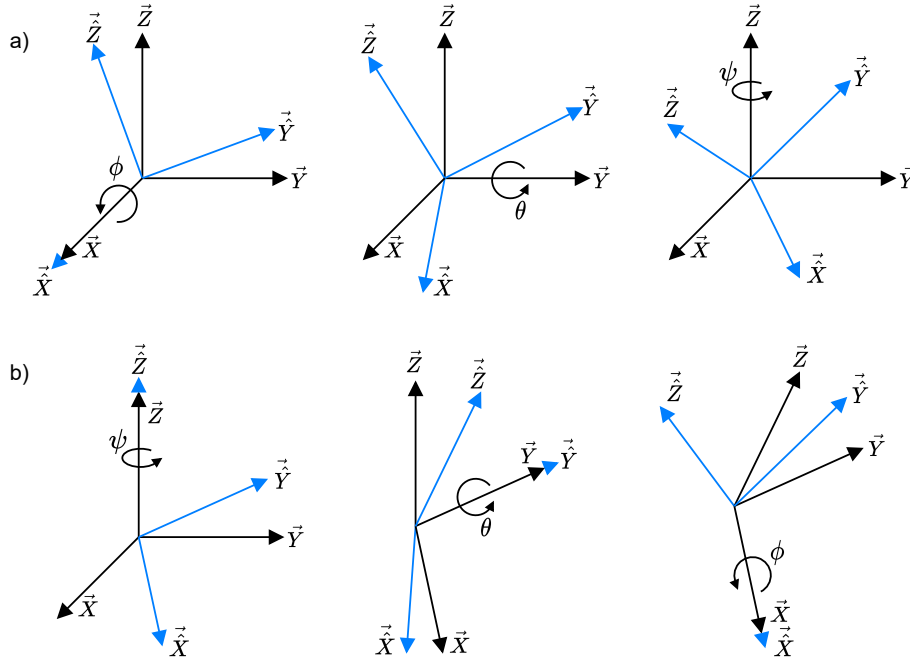
$$\vec{\mathbf{e}}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{\mathbf{e}}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \vec{\mathbf{e}}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (13)$$

However, there is not a single unique way to represent a sequence of Euler rotations. In fact, there are a total of 12 possible unique sequences:

$$\begin{aligned} &XYX, XZX, YZY, YXY, ZXZ, ZYZ, \\ &XYZ, XZY, YZX, YXZ, ZXY, ZYX. \end{aligned} \quad (14)$$

If we take the ZYX sequence as an example, it means we will rotate our frame about the Z-axis first, followed by a rotation about the Y-axis, and finally about the X-axis. However, we will obtain the same final orientation if we consider the XYZ sequence, although the order of the operations is different. Although these two sequences lead to the same orientation, they express distinct behavior, as we can see in Figure 5.

Figure 5: All rotations are executed from left to right where a) represents a XYZ rotation and b) represents a ZYX rotation.



Source: Adapted from (CRAIG, 2005, p. 42-44)

As stated before, the two sequences in Figure 5 lead to the same orientation. However, in item a), we perform rotations in a fixed reference frame, whereas the transformation occurs in a moving frame in item b). In general, the former is applied to aircraft orientation and the latter in attitude control. Since we have 12 possible sequences, we can build several DCMs, but as a demonstration, we will construct a DCM from the XYZ sequence:

$$\mathbf{A} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \quad (15)$$

And for the inverse problem, that is, the extraction of the Euler angles from a given matrix, we can use the formulas given by Equation (16). However, for $\theta = \pm 90^\circ$ the solution degenerates, causing ψ and θ to move in the same fashion. This problem, referred to as *gimbal lock*, causes the loss of one degree of freedom in a three-dimensional

rotation, constraining the system into rotating in a degenerated two-dimensional space. That is why one should be careful when using Euler angles representations.

$$\theta = \arctan2 \left(-\mathbf{A}_{31}, \sqrt{\mathbf{A}_{11}^2 + \mathbf{A}_{21}^2} \right) \quad (16a)$$

$$\psi = \arctan2 \left(\frac{\mathbf{A}_{21}}{\cos\theta}, \frac{\mathbf{A}_{11}}{\cos\theta} \right) \quad (16b)$$

$$\phi = \arctan2 \left(\frac{\mathbf{A}_{32}}{\cos\theta}, \frac{\mathbf{A}_{33}}{\cos\theta} \right) \quad (16c)$$

2.3 THREE-AXIS ATTITUDE DETERMINATION

As previously stated, if a spacecraft needs to know its orientation, some prior information must be gathered by sensors. These data are returned as 3-Dimensional unit vectors, representing the pointing direction of an object of interest concerning the body frame. Therefore, our goal is to find the attitude deviation from this frame to the desired frame expressed in some rotation parametrization. Several algorithms were developed to tackle this problem, such as *Quaternion ESTimator* (QUEST), *Second Estimator of the Optimal Quaternion* (ESOQ2), q-Method, *Singular Value Decomposition* (SVD) based, and others. Nowadays, the most widely used are based on a least-square approximation proposed by Grace Wahba in (WAHBA, 1965).

2.3.1 Wahba's Problem

The data collected by the satellite's onboard sensors are known as measurement or observation vectors. It is possible to obtain two orientation parameters from a given measurement vector. However, three orientation parameters are necessary in order to determine an attitude. Thus, it will lead to an underdetermined problem if only one vector ($\mathbf{N} = 1$) is used because the number of parameters is less than the minimum required. On the other hand, the problem will be overdetermined if two measurement vectors or more ($\mathbf{N} \geq 2$) are used since four orientation parameters will be used. That is, too many variables.

Following this reasoning, some measurements are needed to determine a proper attitude, such that $1 < \mathbf{N} < 2$. However, this is a problem since it is not possible to obtain a non-natural number of measurements. Therefore, a set of measurement vectors are needed to estimate the proper orientation. Let $\vec{\mathbf{r}}_{\mathbf{i}}$ be the reference vectors and $\vec{\mathbf{b}}_{\mathbf{i}}$ be the measurement vectors, such that $\vec{\mathbf{r}}_{\mathbf{i}} \in \{\mathbb{R}^{3 \times 1} \mid \mathbf{i} = 1, 2, \dots, n\}$ and $\vec{\mathbf{b}}_{\mathbf{i}} \in \{\mathbb{R}^{3 \times 1} \mid \mathbf{i} = 1, 2, \dots, n\}$. These vectors are chosen to be unitary because their length has no relevance to attitude determination and a unitary length simplifies the expressions. The objective is to determine a matrix \mathbf{A} that rotates the reference frame to the spacecraft body frame, and it can be

interpreted as a minimization problem. The optimal estimate of \mathbf{A} is defined to be the matrix that minimizes the following loss function:

$$L(\mathbf{A}) = \frac{1}{2} \sum_{i=1}^n \mathbf{a}_i \|\vec{\mathbf{b}}_i - \mathbf{A} \vec{\mathbf{r}}_i\|^2, \text{ such that } n \geq 2 \quad (17)$$

where $\|\cdot\|$ stands for Euclidean norm and \mathbf{a}_i are weights that specify how accurate their corresponding body vectors $\vec{\mathbf{b}}_i$ are, such that $\sum_{i=1}^n \mathbf{a}_i = 1$. This cost function can be rewritten in a more convenient way:

$$L(\mathbf{A}) = \lambda_0 - \text{tr}(\mathbf{A}\mathbf{B}^T) \quad (18)$$

with

$$\lambda_0 = \sum_{i=1}^n \mathbf{a}_i \quad (19)$$

and the attitude profile matrix \mathbf{B} given by

$$\mathbf{B} = \sum_{i=1}^n \mathbf{a}_i \vec{\mathbf{b}}_i \vec{\mathbf{r}}_i^T \quad (20)$$

Algorithms for Wahba's problem can return either an attitude matrix or a quaternion representation. Some are more computation efficient than others, but the ones that solve for quaternions are usually faster and more useful in practice.

2.3.2 Solutions for Wahba's Problem

2.3.2.1 Davenport's q-Method

This algorithm, proposed by Paul B. Davenport in (DAVENPORT, 1968), was the first useful solution for Equation (17). This solution used quaternions as an answer to Wahba's cost function, which could be rewritten as follows:

$$L(\vec{\mathbf{q}}) = \vec{\mathbf{q}}^T \mathbf{K} \vec{\mathbf{q}} \quad (21)$$

This derivation was formally introduced by James E. Keat (KEAT, 1977), where the 4×4 matrix \mathbf{K} is given by Equation (22) and $\vec{\mathbf{z}}$ by Equation (23).

$$\mathbf{K} = \begin{bmatrix} \mathbf{S} - \text{tr}(\mathbf{B})\mathbf{I}_3 & \vec{\mathbf{z}} \\ \vec{\mathbf{z}}^T & \text{tr}(\mathbf{B}) \end{bmatrix} \quad (22)$$

$$\vec{\mathbf{z}} = \begin{bmatrix} \mathbf{B}_{23} - \mathbf{B}_{32} \\ \mathbf{B}_{31} - \mathbf{B}_{13} \\ \mathbf{B}_{12} - \mathbf{B}_{21} \end{bmatrix} \quad (23)$$

where $\mathbf{S} = \mathbf{B} + \mathbf{B}^T$. The unit norm constraint $\vec{\mathbf{q}}^T \vec{\mathbf{q}} = 1$ can be added to Equation (21) since the goal is to find a rotation quaternion. Therefore, the Lagrange multiplier λ can be used to solve this constrained optimization problem, leading to the new gain function $\mathbf{g}(\vec{\mathbf{q}})$ as follows:

$$\mathbf{g}(\vec{\mathbf{q}}) = \vec{\mathbf{q}}^T \mathbf{K} \vec{\mathbf{q}} - \lambda \vec{\mathbf{q}}^T \vec{\mathbf{q}} \quad (24)$$

$$\mathbf{K} \vec{\mathbf{q}} = \lambda \vec{\mathbf{q}} \quad (25)$$

where the derivative of Equation (24) leads to an optimal solution that satisfies Equation (25). Hence, the solution for the Wahba's optimization problem can be found as the largest eigenvalue of \mathbf{K} and its corresponding eigenvector. Worth mentioning that this algorithm does not have a unique solution if the two largest eigenvalues of \mathbf{K} are equal, which means the input data is not sufficient to determine a unique attitude. Nonetheless, the q-Method algorithm still is very robust.

2.3.2.2 SVD based

Introduced by F. Landis Markley in (MARKLEY, 1988), this method proposes the singular value decomposition of the attitude profile matrix \mathbf{B} , Equation (20), which is defined as follows:

$$\mathbf{B} = \mathbf{U} \mathbf{S} \mathbf{V}^T \quad (26)$$

where \mathbf{U} and \mathbf{V} are orthogonal matrices and $\mathbf{S} = \text{diag}(s_1, s_2, s_3)$, such that the singular values are placed in descending order of importance and are greater than zero, that is, $s_1 \geq s_2 \geq s_3 \geq 0$. Then substituting Equation (26) into Equation (18) and doing some matrix manipulation we get:

$$L(\mathbf{A}) = \lambda_0 - \text{tr}(\mathbf{U}^T \mathbf{A} \mathbf{V} \text{diag}(s_1, s_2, s_3)) \quad (27)$$

The loss is minimized, with the constraint $\det(\mathbf{A}) = 1$, when the trace gets maximized by

$$\mathbf{U}^T \mathbf{A}_{opt} \mathbf{V} = \text{diag}(1, 1, \det(\mathbf{U}) \det(\mathbf{V})) \quad (28)$$

resulting in the following optimal rotation matrix

$$\mathbf{A}_{opt} = \mathbf{U} \text{diag}(1, 1, \det(\mathbf{U}) \det(\mathbf{V})) \mathbf{V}^T \quad (29)$$

The uniqueness of the solution relies on the rank of the \mathbf{B} matrix, that is, it depends on the number of non-zero singular values of the attitude profile matrix. The result is not unique if the rank of \mathbf{B} is less than two since we need at least two independent vectors

to define a proper rotation between frames. This method, although very robust, has not been widely used in practice because of its computational burden.

2.3.2.3 QUEST

Unfortunately, the existing algorithms at that time were very computationally expensive. The computation time of both largest eigenvalues and eigenvectors of matrix \mathbf{K} demanded a lot from the onboard hardware. Thus, trying to solve this problem, a novel algorithm was proposed by Malcolm D. Shuster (SHUSTER, M., 1978) and further detailed in (SHUSTER, M. D.; OH, 1981). It was presented the first analytic formula of the characteristic polynomial of the matrix \mathbf{K} , which is a fourth-degree polynomial as follows:

$$\begin{aligned} \lambda^4 - (a + b)\lambda^2 - c\lambda + (ab + c\sigma - d) &= 0 \\ a = \sigma^2 - \kappa \quad b = \sigma^2 + \vec{\mathbf{z}}^T \vec{\mathbf{z}} \\ c = \Delta + \vec{\mathbf{z}}^T \mathbf{S} \vec{\mathbf{z}} \quad d = \vec{\mathbf{z}}^T \mathbf{S}^2 \vec{\mathbf{z}} \end{aligned} \quad (30)$$

where $\sigma = \text{tr}(\mathbf{B})$, $\kappa = \text{tr}(\text{adj}(\mathbf{S}))$, and $\Delta = \det(\mathbf{S})$. The Newton-Raphson iteration, applied to Equation (30), gives us λ_{max} if we use $\lambda_0 = 1$ as initial guess, since λ_{max} is known to be very close to one. In fact, a single iteration is generally sufficient. Still, for a star tracker scenario, where all weights \mathbf{a}_i from Equation (17) are equal, we can get an acceptable accuracy with no iteration.

The optimal quaternion is given by

$$\vec{\mathbf{q}}_{opt} = \frac{1}{\sqrt{\gamma^2 + |\vec{\mathbf{x}}|^2}} \begin{bmatrix} \vec{\mathbf{x}} \\ \gamma \end{bmatrix} \quad (31)$$

where

$$\begin{aligned} \vec{\mathbf{x}} &= (\alpha \mathbf{I}_3 + \beta \mathbf{S} + \mathbf{S}^2) \vec{\mathbf{z}} \\ \gamma &= \alpha(\lambda_{max} + \sigma) - \det(\mathbf{S}) \\ \alpha &= \lambda_{max}^2 - \sigma^2 + \kappa \end{aligned} \quad (32)$$

In practice, depending on the input vectors, the Equation (30) can lead to *Not a Number* (NaN) values during Newton-Raphson iteration. Therefore, using the characteristic equation defined in equation 4 from (MORTARI, 2000) leads to a more numerically stable iteration. However, the optimal quaternion is not defined by Equation (31) if $\gamma^2 + |\vec{\mathbf{x}}|^2 = 0$, which means $\gamma = (\vec{\mathbf{q}}_{opt})_4 = 0$ and the rotational angle is about 180° . A method of sequential rotations was proposed by the same authors to handle this singularity case. The goal is to solve the attitude problem with respect to a rotated reference frame

\mathcal{R}_k , where this new frame is rotated regarding the original frame \mathcal{R} by an angle of 180° about one of the coordinate axes $k \in \{1, 2, 3\}$.

To know whether a sequential rotation is necessary, we must check the absolute value \mathbf{q}_4 of the computed quaternion. According to (MARKLEY; MORTARI, 1999), a $\gamma \geq 0.1$ is sufficient to avoid loss of significance in the computation. Otherwise, a rotation of 180° about the \mathbf{x} -axis is performed and then the γ is checked again. If the condition fails, this step is executed again but for the remaining axes, raising an error if all verifications fail.

A rotation of the reference frame \mathcal{R} is straightforward to implement. If we want to rotate this frame about the \mathbf{x} -axis ($k = 1$), we just need to change the signs of the $k = \{2, 3\}$ columns of the \mathbf{B} matrix. If we want to turn it about the \mathbf{y} -axis, we just need to change the signs of the $k = \{1, 3\}$ columns of \mathbf{B} , and so on. When an acceptable quaternion is found, we must undo the latest rotation. We can achieve this by multiplying the optimal quaternion by the rotating axis. For example, if the \mathbf{B} matrix was rotated about the \mathbf{z} -axis ($k = 3$), then

$$\vec{\mathbf{q}}_{undone} = [\mathbf{q}_{opt,1}, \mathbf{q}_{opt,2}, \mathbf{q}_{opt,3}, \mathbf{q}_{opt,4}]^T \otimes [0, 0, 1, 0]^T \quad (33)$$

where \otimes stands for the quaternion multiplication operation. Since these sequential rotations are executed one axis at a time, this process can be repeated up to three times in the worst scenario case. Still, QUEST is one of the fastest algorithms to compute an attitude rotation. And, according to (MARKLEY; MORTARI, 2000) the characteristic equation is one of the worst ways to find eigenvalues, so QUEST is less robust than q-Method and SVD.

2.3.2.4 ESOQ2

Despite QUEST being one of the fastest algorithms, the need for faster ones was still evident. It would enable the control system to obtain the attitude information at a higher rates. To address this issue, Daniele Mortari (MORTARI, 2000) proposed the following algorithm. It starts from Equation (25), which is equivalent to the following equations:

$$[(\lambda_{max} + \text{tr}(\mathbf{B}))\mathbf{I}_3 - \mathbf{S}]\mathbf{q}_{1-3} = \mathbf{q}_4 \vec{\mathbf{z}} \quad (34a)$$

$$(\lambda_{max} - \text{tr}(\mathbf{B}))\mathbf{q}_4 = \mathbf{q}_{1-3}^T \vec{\mathbf{z}} \quad (34b)$$

Substituting (5) into Equations (34a) and (34b) gives

$$[(\lambda_{max} + \text{tr}(\mathbf{B}))\mathbf{I}_3 - \mathbf{S}]\sin(\phi/2) \vec{\mathbf{e}} = \vec{\mathbf{z}} \cos(\phi/2) \quad (35a)$$

$$(\lambda_{max} - \text{tr}(\mathbf{B}))\cos(\phi/2) \vec{\mathbf{e}} = \vec{\mathbf{z}}^T \sin(\phi/2) \vec{\mathbf{e}} \quad (35b)$$

Multiplying Equation (35a) by $(\lambda_{max} - \text{tr}(\mathbf{B}))$ and substituting into Equation (35b) gives

$$\sin(\phi/2)\mathbf{M}\vec{\mathbf{e}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (36)$$

where \mathbf{M} is given by Equation (37) and the λ_{max} can be obtained by using Newton-Raphson iteration in the same way as QUEST.

$$\mathbf{M} = (\lambda_{max} - \text{tr}(\mathbf{B}))[(\lambda_{max} + \text{tr}(\mathbf{B}))\mathbf{I}_3 - \mathbf{S}] - \vec{\mathbf{z}}\vec{\mathbf{z}}^T = \begin{bmatrix} \mathbf{m}_a & \mathbf{m}_x & \mathbf{m}_y \\ \mathbf{m}_x & \mathbf{m}_b & \mathbf{m}_z \\ \mathbf{m}_y & \mathbf{m}_z & \mathbf{m}_c \end{bmatrix} \quad (37)$$

However, if $(\lambda_{max} - \text{tr}(\mathbf{B}))$ and $\vec{\mathbf{z}}$ are close to zero, then a singularity case arises, indicating that $\phi \rightarrow 0$. Similar to QUEST, we can use sequential rotations to avoid this issue. However, instead of being applied after the quaternion be computed, we can know in advance if a sequential rotation will be needed. No rotation is executed if $\text{tr}(\mathbf{B})$ holds the minimum value among $\{\mathbf{B}_{11}, \mathbf{B}_{22}, \mathbf{B}_{33}, \text{tr}(\mathbf{B})\}$, whereas a rotation about the k th axis is performed if \mathbf{B}_{kk} is the minimum.

From Equation (36), we know that all column vectors of \mathbf{M} are perpendicular to $\vec{\mathbf{e}}$. Therefore, the optimal principal axis can be calculated as a cross-product between two columns of \mathbf{M} , leading us to three possible vectors as follows:

$$\vec{\mathbf{y}}_1 = \mathbf{m}_2 \times \mathbf{m}_3 = [\mathbf{m}_b\mathbf{m}_c - \mathbf{m}_z^2, \mathbf{m}_y\mathbf{m}_z - \mathbf{m}_x\mathbf{m}_c, \mathbf{m}_x\mathbf{m}_z - \mathbf{m}_y\mathbf{m}_b]^T \quad (38a)$$

$$\vec{\mathbf{y}}_2 = \mathbf{m}_3 \times \mathbf{m}_1 = [\mathbf{m}_y\mathbf{m}_z - \mathbf{m}_x\mathbf{m}_c, \mathbf{m}_a\mathbf{m}_c - \mathbf{m}_y^2, \mathbf{m}_x\mathbf{m}_y - \mathbf{m}_z\mathbf{m}_a]^T \quad (38b)$$

$$\vec{\mathbf{y}}_3 = \mathbf{m}_1 \times \mathbf{m}_2 = [\mathbf{m}_x\mathbf{m}_z - \mathbf{m}_y\mathbf{m}_b, \mathbf{m}_x\mathbf{m}_y - \mathbf{m}_z\mathbf{m}_a, \mathbf{m}_a\mathbf{m}_b - \mathbf{m}_x^2]^T \quad (38c)$$

Because \mathbf{M} is singular, all $\vec{\mathbf{y}}_i$ are parallel. Hence, all are parallel to the rotation axis $\vec{\mathbf{e}}$. Thus, we should choose the rotation axis $\vec{\mathbf{y}}_i$ with the maximum norm, which give us

$$\vec{\mathbf{q}}_{opt} = \frac{1}{\sqrt{(\lambda_{max} - \text{tr}(\mathbf{B}))\vec{\mathbf{y}}^2 + (\vec{\mathbf{z}} \cdot \vec{\mathbf{y}})^2}} \begin{bmatrix} (\lambda_{max} - \text{tr}(\mathbf{B}))\vec{\mathbf{y}} \\ \vec{\mathbf{z}} \cdot \vec{\mathbf{y}} \end{bmatrix} \quad (39)$$

In the same way as QUEST, the prior rotation must be undone by using the Equation (33) after computing the optimal quaternion. Worth to mention that all solutions for quaternion presented so far must be inverted in order to work, that is, $\vec{\mathbf{q}}_{valid} = \vec{\mathbf{q}}_{opt}^{-1} = [-\mathbf{q}_1, -\mathbf{q}_2, -\mathbf{q}_3, \mathbf{q}_4]$.

3 RELATED WORK

Some recent works using *Artificial Neural Network* (ANN) have been developed to tackle attitude determination problems. However, most of them only consider dynamic attitude determination, that is, they depend on previous estimations to predict new orientations. Some of them use ANN along with Kalman filters (CHIANG et al., 2009; RAMBACH et al., 2016), and others use *Recurrent Neural Network* (RNN) architectures to estimate the new attitude (ESFAHANI et al., 2019; WEBER et al., 2020). However, regression on rotation representations using ANN is not new. Actually, there is an extensive field of study covering this area. These approaches are mainly used in computer vision and robotics, where the system must infer objects perspectives or joint angles.

In the work proposed by (XIANG et al., 2017), they try to estimate the rigid transformation from the object coordinate frame \mathcal{O} to the camera coordinate frame \mathcal{C} , given an input image containing a collection of objects. The main task is to find their poses (location and orientation), represented by a 3D rotation \mathbf{R} and a 3D translation \mathbf{T} . These rotations can be rewritten as $\mathbf{R}(\vec{\mathbf{q}})$ since quaternions parametrize them. The mean squared error loss function minimizes the error considering the distance between a set of 3D model points $\vec{\mathbf{x}} \in \mathcal{O}$ rotated by both the groundtruth rotation $\mathbf{R}(\vec{\mathbf{q}})$ and the predicted rotation $\mathbf{R}(\vec{\hat{\mathbf{q}}})$. However, as mentioned by the authors, the network perform poorly when the rotation angles are close to 0° and 180° .

In (DO et al., 2018), they use an architecture called *Mask Region Based Convolutional Neural Networks* (Mask R-CNN) developed by (HE et al., 2017), which takes an RGB image containing several objects as input, returning their classes, segmentation masks, and bounding boxes. They added a fourth output on top of this architecture, which tries to estimate the orientation and localization of each object. This output layer is a 4-Dimensional vector where the first three scalars represent the rotation, and the last one represents the translation. However, they use Lie algebra $so(3)$ to describe the rotation instead of using representations, such as Euler angles and quaternions, because a skew-symmetric matrix of an arbitrary element of $so(3)$ can be mapped to the *Special Orthogonal Group* (SO(3)) through exponential mapping. Hence, they only need to regress a vector $\vec{\mathbf{x}} \in \mathbb{R}^3$ to define a proper rotation.

Point clouds can be interpreted as 3D vectors that describe a shape or object in some arbitrary coordinate frame. Hence, this data type can provide a lot of helpful information, such as depth, volume, orientation, and location. Trying to explore this type of information, (GAO et al., 2018) proposed a novel approach to address the 6D pose estimation problem: Instead of using RGB images, the system uses point cloud segments as input. The proposed neural network is an adaptation of the PointNet (QI et al., 2017) architecture, which takes a set of point clouds as input and returns a rotation representation that best describes the orientation of the input data. The geodesic distance

defined in Equation (40) directly measures the magnitude of rotation difference between the ground truth and the estimated rotation, where ϕ represents the angle error, \mathbf{R} the ground truth rotation and $\hat{\mathbf{R}}$ the predicted rotation.

$$\phi(\mathbf{R}, \hat{\mathbf{R}}) = \arccos\left(\frac{\text{tr}(\mathbf{R}\hat{\mathbf{R}}^T) - 1}{2}\right) \quad (40)$$

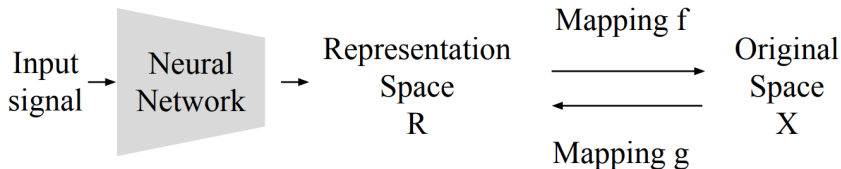
However, as pointed out by (ZHOU et al., 2019), 3D and 4D representations are not ideal for network regression when the entire rotation space is required. They demonstrated those parametrizations are discontinuous, that is, they do not express a smooth transition for certain angles. Taking the Euler angles representation as an example: If we consider the rotation matrix \mathbf{M} , as shown in Equation (41), which describes a rotation about some arbitrary axis, we will get an ambiguity for the identity rotation. This means that we will get multiple valid answers for the same orientation: an angle of $\mathbf{0}$ and an angle of $\mathbf{2\pi}$.

$$\mathbf{M} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (41)$$

The quaternions representation is also discontinuous, despite being vastly used in many areas. To demonstrate this, let us consider a function $\mathbf{g}(\cdot)$, defined by Equation (7d), that maps $\mathbf{M} \in SO(3)$ to the quaternions, which are members of the Euclidean space \mathbb{R}^4 . We can see that as $\theta \rightarrow 180^\circ$, the resulting quaternion becomes the null vector $\mathbf{g}(\mathbf{M}) = [0, 0, 0, 0]$. Hence, the limit goes to infinity if we try to normalize it. Therefore, it is not a proper representation for $SO(3)$.

Trying to find a valid continuous representation for $SO(3)$, (ZHOU et al., 2019) proposed a new method. They assumed that a *representation space* \mathcal{R} produced by a neural network could be mapped into the *original space* $\mathcal{X} \in SO(3)$ through a function $\mathbf{f} : \mathcal{R} \rightarrow \mathcal{X}$ and mapped back through a function $\mathbf{g} : \mathcal{X} \rightarrow \mathcal{R}$, as depicted in Figure 6.

Figure 6: **Representation mapping.**



Source: (ZHOU et al., 2019)

The function \mathbf{f} is a mathematical function used as part of the forward pass of the network at both training and inference time. For 6D representations, this function is guaranteed to return an orthogonal 3x3 matrix, which we can apply the loss function. This mapping function is defined as follows for 6D representation:

$$\mathbf{f} \left(\begin{bmatrix} | & | \\ \vec{\mathbf{a}}_1 & \vec{\mathbf{a}}_2 \\ | & | \end{bmatrix} \right) = \begin{bmatrix} | & | & | \\ \vec{\mathbf{b}}_1 & \vec{\mathbf{b}}_2 & \vec{\mathbf{b}}_3 \\ | & | & | \end{bmatrix} \quad (42a)$$

$$\vec{\mathbf{b}}_{\mathbf{i}} = \begin{bmatrix} N(\vec{\mathbf{a}}_1) & \text{if } \mathbf{i} = 1 \\ N(\vec{\mathbf{a}}_2 - (\vec{\mathbf{b}}_1 \cdot \vec{\mathbf{a}}_2) \vec{\mathbf{b}}_1) & \text{if } \mathbf{i} = 2 \\ \vec{\mathbf{b}}_1 \times \vec{\mathbf{b}}_2 & \text{if } \mathbf{i} = 3 \end{bmatrix}^{\mathbf{T}} \quad (42b)$$

where $N(\cdot)$ stands for normalization function, \times for cross product, \cdot for dot product, and the column vectors $\vec{\mathbf{a}}_1$ and $\vec{\mathbf{a}}_2$ are vectors from \mathcal{R} estimated by the network. Similar to (GAO et al., 2018), they also used a PointNet architecture and point clouds as input to the system.

The algorithms presented in this chapter share similar goals despite not being directly related to attitude determination: they estimate a rotation representation that leads a collection of vectors in \mathbb{R}^3 from one coordinate frame to another with the lowest error possible. Thus, we have taken advantage of these approaches to address the static attitude determination problem in this work.

4 EXPERIMENTS

This chapter will detail how we developed the dataset, network architecture, training, and test procedures. We produced the software using Tensorflow 2 framework due to its ease of use and detailed documentation. Later on in the chapter, we will display some results that we got after training the models.

4.1 DATASET

The traditional algorithms, as explained before, were designed to "figure out" the best attitude representation given a set of unit observation vectors and their respective weights. In essence, these vectors are random, and the only requirement is that they must be unitary and have a boresight axis, which expresses the direction of the observed object.

Following this reasoning, we built the dataset given the number of samples and observations per sample. First, we generated random unit vectors from a uniform distribution ranging from 0 to 1 with arbitrary boresight axes, forming an extensive array with a shape $(\mathcal{S}, \mathcal{O}, 3)$, where \mathcal{S} stands for the number of samples, \mathcal{O} the number of observation vectors, and the last dimension represents the number of axes. Since we are dealing with components in \mathbb{R}^3 , we set the final dimension to 3. This first collection of vectors holds the reference vectors $\vec{\mathbf{r}}$.

After that, we generated random rotation matrices \mathbf{A}_{true} by using Equation (3), where we drew the angles from a uniform distribution ranging from $-\pi$ rad to π rad. We drew the rotation axis vectors from another uniform distribution and normalized them. These random DCMs with a shape $(\mathcal{S}, 3, 3)$ served as the ground truth attitude matrices. Next, the reference vectors were rotated by each attitude matrix, generating the body vectors $\vec{\mathbf{b}}$ with a shape $(\mathcal{S}, \mathcal{O}, 3)$ as follows:

$$\vec{\mathbf{b}}_{\mathbf{i}} = \mathbf{A}_{\text{true},\mathbf{i}} \vec{\mathbf{r}}_{\mathbf{i}} + \vec{\mathbf{n}}_{\mathbf{i}} \quad (43)$$

where $\vec{\mathbf{n}}_{\mathbf{i}}$ is a vector of measurement errors drawn from a zero-mean Gaussian distribution $\mathcal{N}(0, \sigma_{\mathbf{i}})$ whose standard deviations $\sigma_{\mathbf{i}}$ were random variables drawn from a uniform distribution ranging from 10^{-6} to 10^{-2} serving as *Additive White Gaussian Noise* (AWGN).

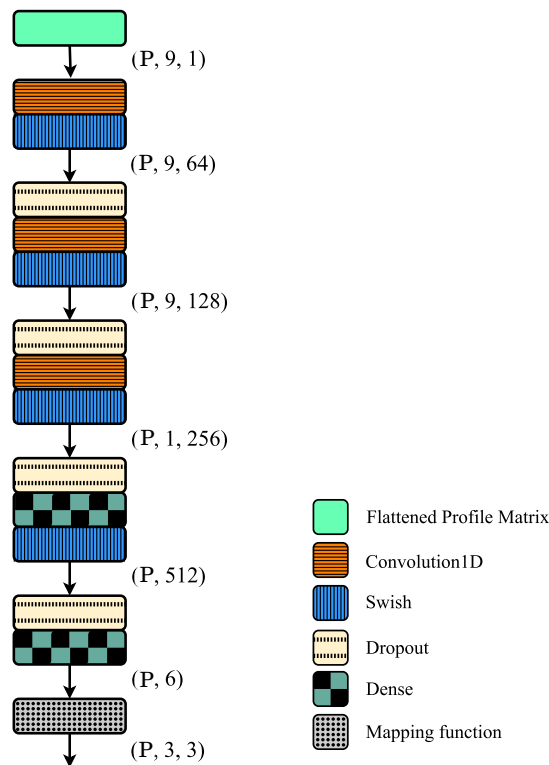
4.2 NEURAL NETWORK ARCHITECTURE

Similar to the works explained in the last chapter, we based our model on the PointNet architecture, whose structure can be seen in Figure 7. First, we did some modifications to the traditional PointNet to improve its performance. In the vanilla architecture, a set of vectors in the original coordinate frame and another set in the target coordinate frame are used as input and fed into the model in pairs. However, in the proposed model, they

were replaced by the attitude profile matrix. In fact, we used the same pairs of vectors to build the profile matrix according to Eq. (20). By flattening this matrix, an array with a shape $(\mathbf{P}, 9, 1)$ is generated, where \mathbf{P} represents the batch size. However, it was considered that all weights \mathbf{a}_i were equal (star tracking scenario) because it led to lower errors when compared to random weights even on the evaluation phase.

After that, we built all 1-dimensional convolutions using a kernel of size 9. However, we maintained the shape of the feature maps in all but the last convolution. As a result, the second dimension of the tensor did not change after applying the convolution operation. Next, we replaced all *Rectified Linear Unit* (ReLU) activation functions with the Swish activation (RAMACHANDRAN et al., 2017) because it turned out that ReLU led to more significant errors during training and evaluation. Finally, as the last operation, we implemented the mapping function \mathbf{f} that maps a latent rotation representation \mathcal{R} to $SO(3)$, as performed by (ZHOU et al., 2019).

Figure 7: Modified PointNet.



Source: Own authorship

We added Dropout (SRIVASTAVA et al., 2014) layers after each activation function. The reasons are twofold: to act as a regularizer to prevent overfitting and measure the model uncertainty. While training, each update to a layer is performed with a different view of the configured layer since the dropout technique randomly drops some neurons with a probability \mathbf{p} at each forward pass. This "forces" the knowledge to be spread out across all neurons equally, thus acting as a regularizer. This can be interpreted as if several neural networks are being trained simultaneously because the model "changes" its

architecture at each forward pass.

However, in (GAL; GHAHRAMANI, 2016), they realized that regular dropout could be interpreted as a Bayesian approximation of a Gaussian process. When applying the dropout both during training and inference, it is possible to analyze it as if we took predictions from many different networks, that is, a Monte Carlo sample from the space of all possible networks. Hence, it is entitled as *Monte Carlo Dropout* (MCD). Therefore, we can gather a distribution of predictions and evaluate the uncertainty of the model.

4.3 TRAINING

We generated a dataset with the total number of samples $\mathcal{S} = 2^{13}$, where we used 30% of it as a test dataset and 4% as a validation dataset. We optimized our model using the AdamW algorithm (LOSHCHILOV; HUTTER, 2017), with a learning rate $\mathbf{lr} = 10^{-4}$ and a small weight decay of 10^{-8} . We also reduced both the \mathbf{lr} and the weight decay by a factor of ten every 500 epochs. We chose this starting value for the \mathbf{lr} because the network obtained higher errors during the training phase for a $\mathbf{lr} = 10^{-3}$.

To train the network, we used the loss function defined by Equation (40). We clipped it to lie in the range $[-1 + \text{eps}, 1 - \text{eps}]$ to avoid numerical issues, where an $\text{eps} = 10^{-7}$ is enough. We used this same equation as a metric along with the Wahba’s loss, defined by Equation (17), where we considered all weights \mathbf{a}_i to be equal during the metric evaluation. We scheduled several training routines assuming different values of \mathcal{O} for each one to check if this could bring some meaningful impact on the model’s robustness. Different models were trained considering a value for \mathcal{O} ranging from 3 up to 7. Every training was performed considering 2000 epochs, with a batch size of 64. Besides that, for each \mathcal{O} , we assumed three different dropout rates: 10%, 15% and 20%. We have chosen this range of values for the dropout rate based on the values used in section 5.1 from the original MCD paper. Furthermore, larger dropout rate values could lead to divergence or require more iterations to converge. Conversely, a minimal dropout rate value would eliminate the Monte-Carlo sampling effectiveness.

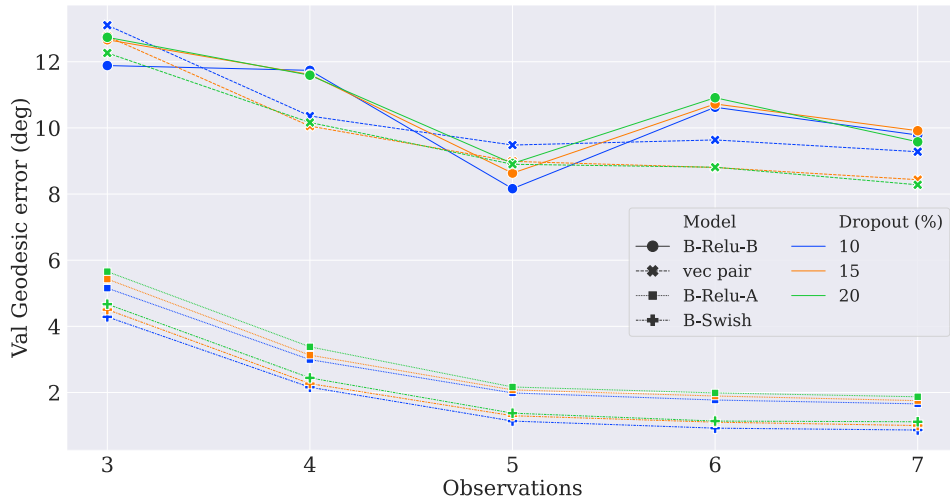
Initially, we trained an earlier version of the architecture presented in Section 4.2, where the pairs of body and reference vectors, concatenated along the last dimension, were defined as the input to the network. We set the number of observation vectors as the kernel size parameter in the convolution layers. We maintained the shape of the feature maps in all except the last convolution. In this assessment, the ReLU activation function was implemented instead of Swish.

We made variations to improve the model since this assessment did not achieve acceptable error values. We replaced the pair of vectors with the attitude profile matrix; thus, the architecture presented in Fig. 7 was formulated. However, considering ReLU instead of Swish. Throughout this thesis, this model will be called B-ReLU. This model

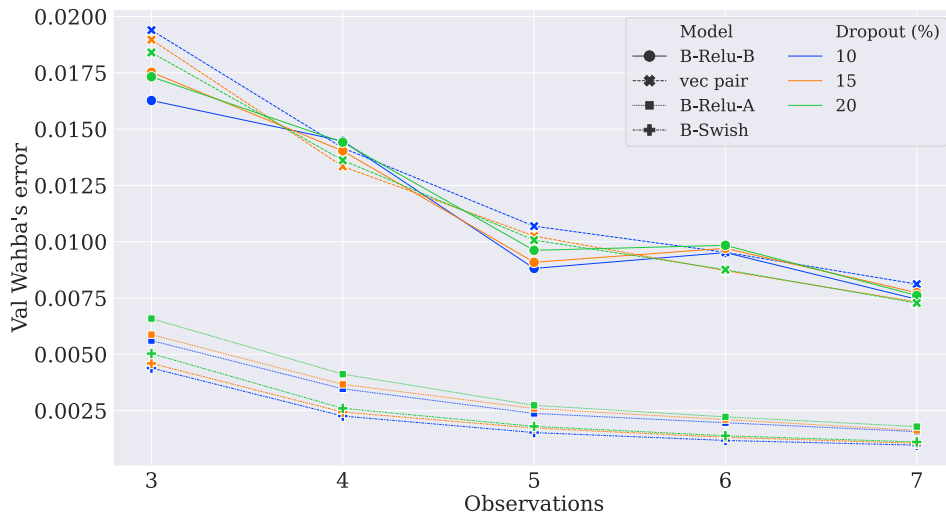
had two versions: B-ReLU-A and B-ReLU-B. The former assumes a star tracking scenario, and the latter considers the actual weights for each pair of inputs when building the attitude profile matrices. We noticed that using the actual weights to build the \mathbf{B} matrix led to poor results; thus, we decided to use the star tracking scenario. Since the Swish activation function presented promising results, according to its original paper, we decided to put it in place of ReLU and check whether we could benefit from it. In this way, we built the final model as explained in Section 4.2, which we have called as B-Swish.

After training the models, we measured their performances over the validation set. We tested the models whose parameters corresponded to the slightest error in Wahba's loss. As we can see in Figure 8, we achieved better results over the validation dataset when we increased the hyperparameter \mathcal{O} . This decrease in error indicated the network captured more insights about the attitude profile matrix since it was using more observation vectors. Furthermore, as we increased the dropout rate, the network tended to get worse results, indicating we were over-penalizing the parameters.

Figure 8: Results over the validation dataset.



(a) Geodesic error in degrees



(b) Wahba's loss

Source: Own authorship

As mentioned before, the network struggled to converge when using pairs of body and reference vectors as inputs. Moreover, when considering the accurate weights to build the attitude profile, we obtained worse results. And as expected, the Swish activation function improved the network performance; thus, we have chosen the structure presented in Figure 7 as the final architecture. However, we still needed to verify which of those B-Swish models was more reliable and if those results reflected the system's expected behavior when presented to unseen scenarios.

4.4 MODEL CHOICE

We still need to do some verifications to validate which of those models is a better candidate. We will follow the same tests as proposed by (MARKLEY, 1993), where

the author proposed twelve test scenarios specified by a set of reference vectors $\vec{\mathbf{r}}_i$ and measurement noise σ_i . The body vectors $\vec{\mathbf{b}}_i$ were constructed following Equation (43), where the \mathbf{A}_{true} is

$$\mathbf{A}_{\text{true}} = \begin{bmatrix} 0.352 & 0.864 & 0.360 \\ -0.864 & 0.152 & 0.480 \\ 0.360 & -0.480 & 0.800 \end{bmatrix} \quad (44)$$

The twelve cases were specified as follows:

Case 1 In this case we have three measurement vectors with measurement noise $\sigma_1 = \sigma_2 = \sigma_3 = 10^{-6}$ rad as follows

$$\vec{\mathbf{r}}_1 = [1, 0, 0]^{\mathbf{T}}, \quad \vec{\mathbf{r}}_2 = [0, 1, 0]^{\mathbf{T}}, \quad \vec{\mathbf{r}}_3 = [0, 0, 1]^{\mathbf{T}} \quad (45)$$

Case 2 Used the same $\vec{\mathbf{r}}_1$ and $\vec{\mathbf{r}}_2$ vectors from **Case 1** with measurement noise $\sigma_1 = \sigma_2 = 10^{-6}$ rad.

Case 3 The same three vectors from **Case 1** but increased the noise to $\sigma_1 = \sigma_2 = \sigma_3 = 0.01$ rad.

Case 4 The same as **Case 2** but increased the noise to $\sigma_1 = \sigma_2 = 0.01$ rad.

Case 5 Used two reference vectors with measurement noise $\sigma_1 = 10^{-6}$ and $\sigma_2 = 0.01$ rad as follows

$$\vec{\mathbf{r}}_1 = [0.6, 0.8, 0]^{\mathbf{T}}, \quad \vec{\mathbf{r}}_2 = [0.8, -0.6, 0]^{\mathbf{T}} \quad (46)$$

Case 6 In this case we have three measurement vectors with measurement noise $\sigma_1 = \sigma_2 = \sigma_3 = 10^{-6}$ rad as follows

$$\vec{\mathbf{r}}_1 = [1, 0, 0]^{\mathbf{T}}, \quad \vec{\mathbf{r}}_2 = [1, 0.01, 0]^{\mathbf{T}}, \quad \vec{\mathbf{r}}_3 = [1, 0, 0.01]^{\mathbf{T}} \quad (47)$$

Case 7 Used the same $\vec{\mathbf{r}}_1$ and $\vec{\mathbf{r}}_2$ vectors from **Case 6** with measurement noise $\sigma_1 = \sigma_2 = 10^{-6}$ rad.

Case 8 The same three vectors from **Case 6** but increased the noise to $\sigma_1 = \sigma_2 = \sigma_3 = 0.01$ rad.

Case 9 The same as **Case 7** but increased the noise to $\sigma_1 = \sigma_2 = 0.01$ rad.

Case 10 Used three reference vectors with measurement noise $\sigma_1 = 10^{-6}$ rad and $\sigma_2 = \sigma_3 = 0.01$ rad as follows

$$\vec{\mathbf{r}}_1 = [1, 0, 0]^{\mathbf{T}}, \quad \vec{\mathbf{r}}_2 = [0.96, 0.28, 0]^{\mathbf{T}}, \quad \vec{\mathbf{r}}_3 = [0.96, 0, 0.28]^{\mathbf{T}} \quad (48)$$

Case 11 Used the same $\vec{\mathbf{r}}_1$ and $\vec{\mathbf{r}}_2$ vectors from **Case 10** with measurement noise $\sigma_1 = 10^{-6}$ rad and $\sigma_2 = 0.01$ rad.

Case 12 Used the same $\vec{\mathbf{r}}_1$ and $\vec{\mathbf{r}}_2$ vectors from **Case 10** with measurement noise $\sigma_1 = 0.01$ rad and $\sigma_2 = 10^{-6}$ rad.

For each case, we built the attitude profile matrix as defined in Equation (20) to serve as input to the neural network, where we considered a star tracker scenario, that is, each observation vector was equally weighted. Next, with the dropout layers enabled, we performed \mathbf{N} forward passes through the network to sample a set of possible outputs. In our tests, we defined $\mathbf{N} = 1000$. After each forward pass, we evaluated the Wahba's error using the Equation (17). We considered the weights given by the test case, defined by Eq. (49b), and we assumed \mathbf{A}_{pred} instead of \mathbf{A}_{true} to measure the true distance. This led to an array of errors with size \mathbf{N} for each test case. At the end of the \mathbf{N} forward passes, we measured the mean error for each test case, as shown in Table 3. The error values are displayed on a logarithmic scale, where the blue entries represent the smallest values.

$$\sigma_{\text{tot}} = \left(\sum_{i=1}^{\mathcal{O}} \frac{1}{\sigma_i^2} \right)^{-1} \quad (49a)$$

$$\mathbf{a}_i = \frac{\sigma_{\text{tot}}}{\sigma_i^2} \quad (49b)$$

Table 3: Wahba's error for each model.

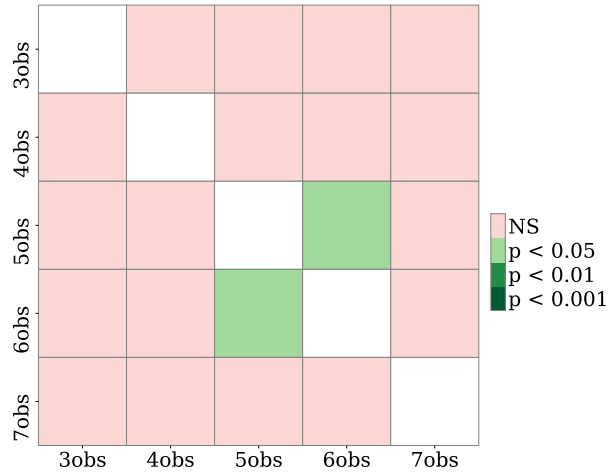
Model Dropout rate (%) Case	3obs			4obs			5obs			6obs			7obs		
	10	15	20	10	15	20	10	15	20	10	15	20	10	15	20
1	-2.03	-1.96	-1.90	-2.04	-2.00	-1.93	-2.17	-2.02	-1.99	-2.10	-2.08	-1.99	-2.14	-2.07	-1.97
2	-1.81	-1.75	-1.68	-1.86	-1.78	-1.73	-1.92	-1.82	-1.76	-1.86	-1.85	-1.76	-1.95	-1.80	-1.76
3	-2.00	-1.94	-1.90	-2.06	-1.98	-1.92	-2.12	-2.02	-1.97	-2.08	-2.05	-1.98	-2.11	-2.06	-1.97
4	-1.80	-1.74	-1.69	-1.86	-1.78	-1.74	-1.92	-1.83	-1.75	-1.85	-1.84	-1.75	-1.91	-1.79	-1.77
5	-1.82	-1.75	-1.69	-1.81	-1.72	-1.71	-1.94	-1.79	-1.78	-1.90	-1.88	-1.79	-1.93	-1.79	-1.74
6	-1.94	-1.87	-1.85	-2.05	-1.97	-1.91	-1.89	-1.87	-1.71	-1.75	-1.78	-1.73	-1.83	-1.77	-1.75
7	-1.77	-1.67	-1.65	-1.87	-1.78	-1.72	-1.70	-1.68	-1.53	-1.58	-1.61	-1.53	-1.65	-1.60	-1.57
8	-1.92	-1.85	-1.83	-2.02	-1.96	-1.87	-1.89	-1.86	-1.69	-1.75	-1.77	-1.74	-1.82	-1.76	-1.74
9	-1.75	-1.68	-1.65	-1.85	-1.78	-1.71	-1.71	-1.69	-1.52	-1.56	-1.60	-1.55	-1.65	-1.60	-1.56
10	-1.95	-1.92	-1.83	-1.88	-1.91	-1.88	-1.92	-1.90	-1.81	-1.73	-1.73	-1.76	-1.73	-1.70	-1.64
11	-1.75	-1.68	-1.65	-1.65	-1.63	-1.62	-1.67	-1.65	-1.65	-1.64	-1.56	-1.54	-1.54	-1.58	-1.54
12	-1.56	-1.53	-1.53	-1.54	-1.58	-1.57	-1.49	-1.64	-1.53	-1.37	-1.44	-1.46	-1.39	-1.46	-1.35

Source: Own authorship

If we only consider the errors displayed in Table 3 to choose the most eligible model, it would still not be possible to notice any meaningful insight. Even though the models trained using four and five observations, and a dropout rate of 10% performed, in general, better than the others. Due to that, we carried out the Friedman test (FRIEDMAN, 1937), a non-parametric statistical test, over the results presented in Table 3 to verify whether there is a significant difference between the models. The Friedman test returned p-values above 0.05 for the models trained with a dropout rate of 15% and 20%, indicating the null hypothesis could not be rejected, that is, the models trained with a dropout rate of 15% and 20% did not present significant difference regarding the number of observation vectors.

However, the null hypothesis was rejected for the models with a dropout rate of 10%. Therefore, we performed the Nemenyi post-hoc analysis (NEMENYI, 1963) concerning those models to find out which of them differs from the others.

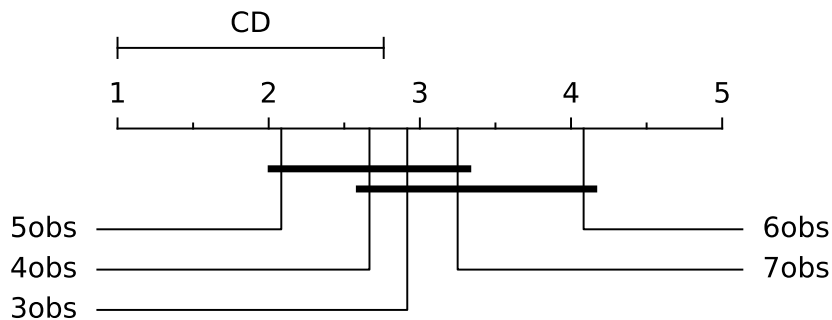
Figure 9: **The relationship of the p-values among all models.**



Source: Own authorship

Based on the heatmap displayed in Figure 9, we noticed that only the models trained with 5 and 6 observation vectors presented some difference. However, this deviation is not relevant because they have close similarities with the other models, that is, they are grouped together because their difference is less than the *Critical Difference* (CD) as we can see in Figure 10. Thus, we could not choose the most appropriate model based on the results presented in Table 3. Hence, we followed a procedure described by (FRANASZEK; CHEOK, 2017), where they evaluated the rotation error concerning the average rotation matrix of the predicted DCMs.

Figure 10: **Critical difference diagram for Nemenyi tests. Bold lines indicate groups of models which are not significantly different (their average ranks differ by less than the $CD = 1.761$).**



Source: Own authorship

Given the set of predicted DCMs $\{\hat{\mathbf{R}}_j \mid j = 1, 2, \dots, \mathbf{N}\}$, it is possible to calculate an

average rotation matrix \mathbf{R}_{avg} , that is, a matrix that represents the mean of a distribution of rotation matrices. This average rotation was found as the orthogonal projection of the matrix $\bar{\mathbf{R}} = \mathbf{N}^{-1} \sum_{j=1}^{\mathbf{N}} \hat{\mathbf{R}}_j$ and can be further reviewed in (MOAKHER, 2002) and (CURTIS et al., 1993), the latter uses a SVD approach to evaluate \mathbf{R}_{avg} . Other interesting approaches are detailed in (SHARF et al., 2010).

Once \mathbf{R}_{avg} is determined, the noise matrices $\Delta\mathbf{R}_j$ can be evaluated. These new matrices hold the orientation differences between each predicted DCM $\hat{\mathbf{R}}_j$ and the average DCM \mathbf{R}_{avg} , which can be found by doing the following matrix multiplication:

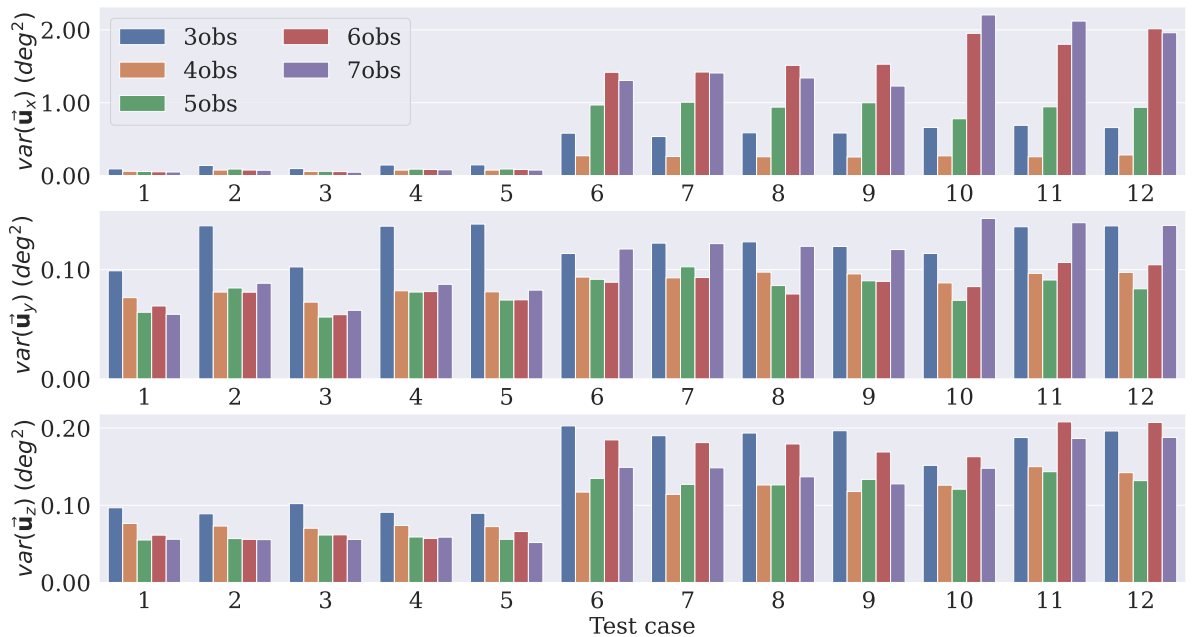
$$\Delta\mathbf{R}_j(\vec{\mathbf{e}}_j, \phi_j) = \mathbf{R}_{\text{avg}}^{\mathbf{T}} \hat{\mathbf{R}}_j \quad (50)$$

where each $\Delta\mathbf{R}_j$ is parametrized by a rotation axis $\vec{\mathbf{e}}_j$ and a rotation angle ϕ_j . After finding the noise matrices, we extracted their respective Euler vectors $\vec{\mathbf{u}}_j = \phi_j \vec{\mathbf{e}}_j$ and calculated the covariance matrix as follows:

$$\mathbf{C}(\vec{\mathbf{u}}) = \frac{1}{\mathbf{N}} [\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \dots, \vec{\mathbf{u}}_{\mathbf{N}}][\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \dots, \vec{\mathbf{u}}_{\mathbf{N}}]^{\mathbf{T}} \quad (51)$$

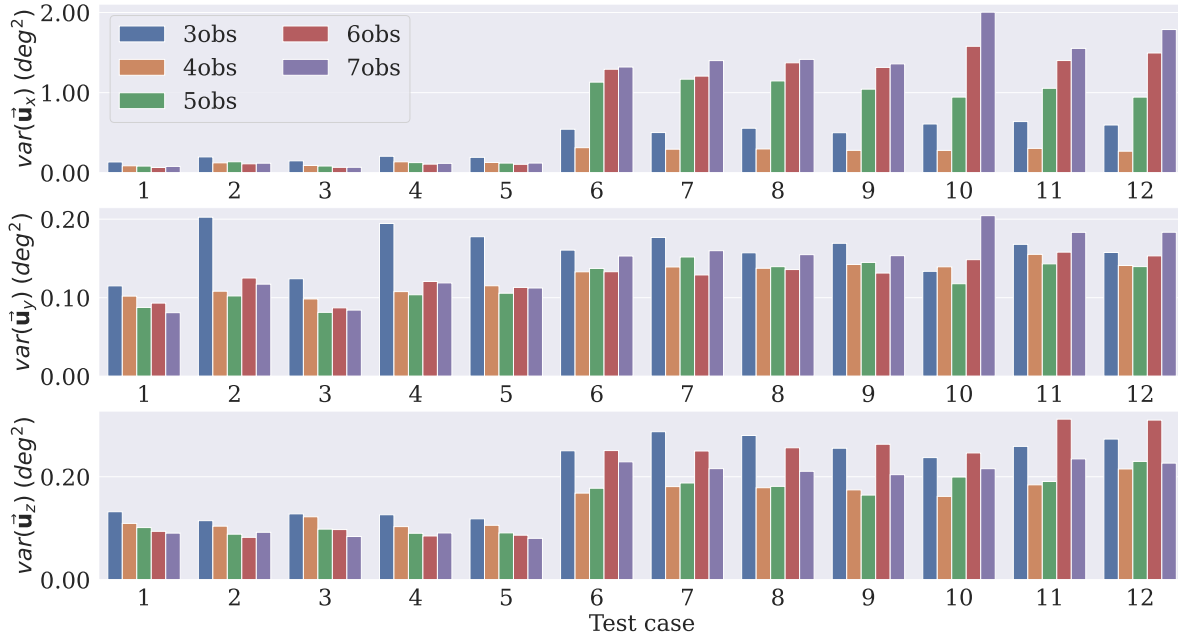
Since $\vec{\mathbf{u}}_j$ is a representation of the difference rotation matrix $\Delta\mathbf{R}_j$, by taking the diagonal entries of $\mathbf{C}(\vec{\mathbf{u}})$ it is possible to obtain the variances in the \mathbf{x} , \mathbf{y} and \mathbf{z} axes of the predicted DCMs around \mathbf{R}_{avg} . This procedure was performed for each test case to measure the uncertainty of the rotation axes predicted by each model. The measured variances, in degrees², for each dropout rate, are displayed in Figures 11, 12 and 13.

Figure 11: **Axes uncertainties for each test case considering a dropout rate of 10%.**



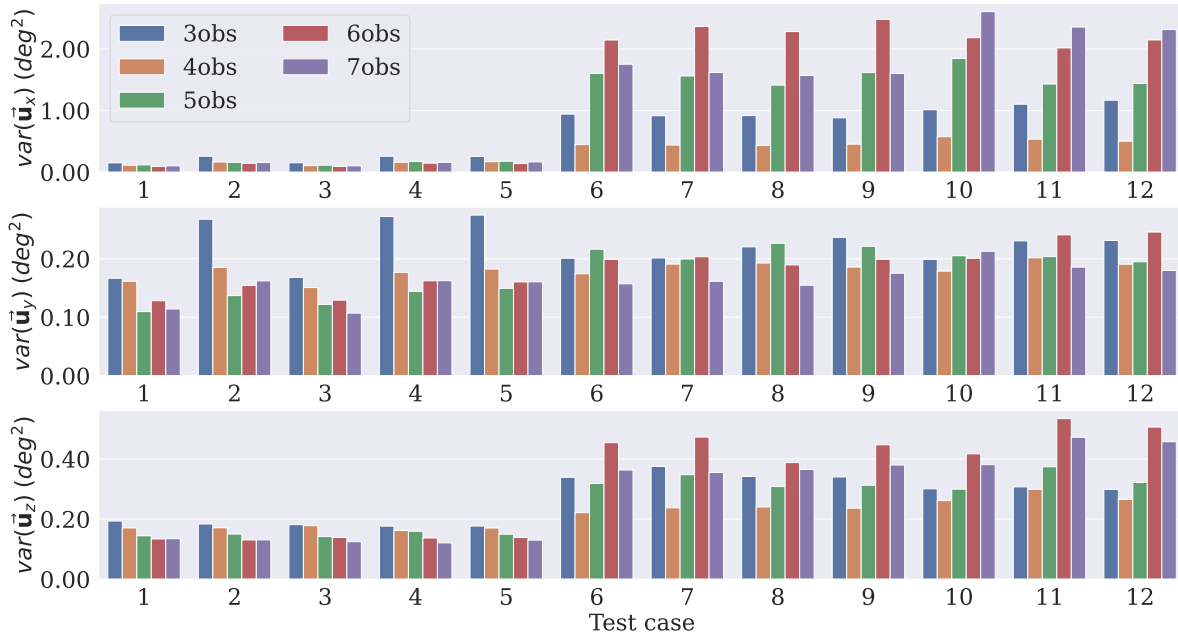
Source: Own authorship

Figure 12: Axes uncertainties for each test case considering a dropout rate of 15%.



Source: Own authorship

Figure 13: Axes uncertainties for each test case considering a dropout rate of 20%.



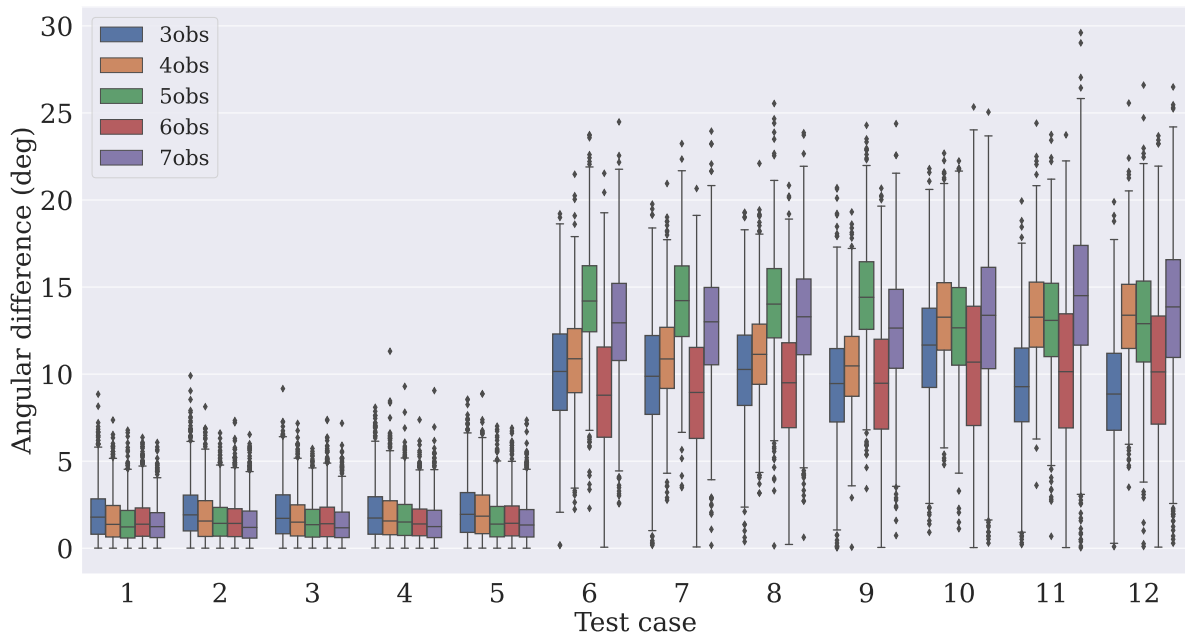
Source: Own authorship

The models with a dropout rate of 10% had lower variance levels in the three axes for the majority of the test cases. Worth mentioning, we tested those models using the same dropout rate \mathbf{p} they were trained with. By doing that, it would be possible

to verify whether, independently of the dropout rate used, the resulting pattern would be maintained. The model trained using four observations obtained smaller variances for the $\vec{\mathbf{u}}_x$ component compared to the others, even though the values were very similar for the $\vec{\mathbf{u}}_y$ and $\vec{\mathbf{u}}_z$ components. Those results led us to choose the model trained using four observation vectors as the best option. It was impossible to use the standard equations of mean and variance to calculate the uncertainty since the experiments deal with directional data. Nonetheless, it was still possible to use those predictions and the covariance approach to estimate the uncertainty because enabling the dropout layer during inference generates different outputs for the same input.

We also measured the distance between the true angle, extracted from Equation 44, and the angles from the predicted attitude matrices \mathbf{A}_{pred} . For each test case, \mathbf{N} predictions were taken, with their angles extracted using Equation 4a. The Figures 14, 15 and 16 display how the absolute angular differences, in degrees, are spread for the three dropout values. We noticed that the dispersion of the differences was narrower for the model using four observation vectors.

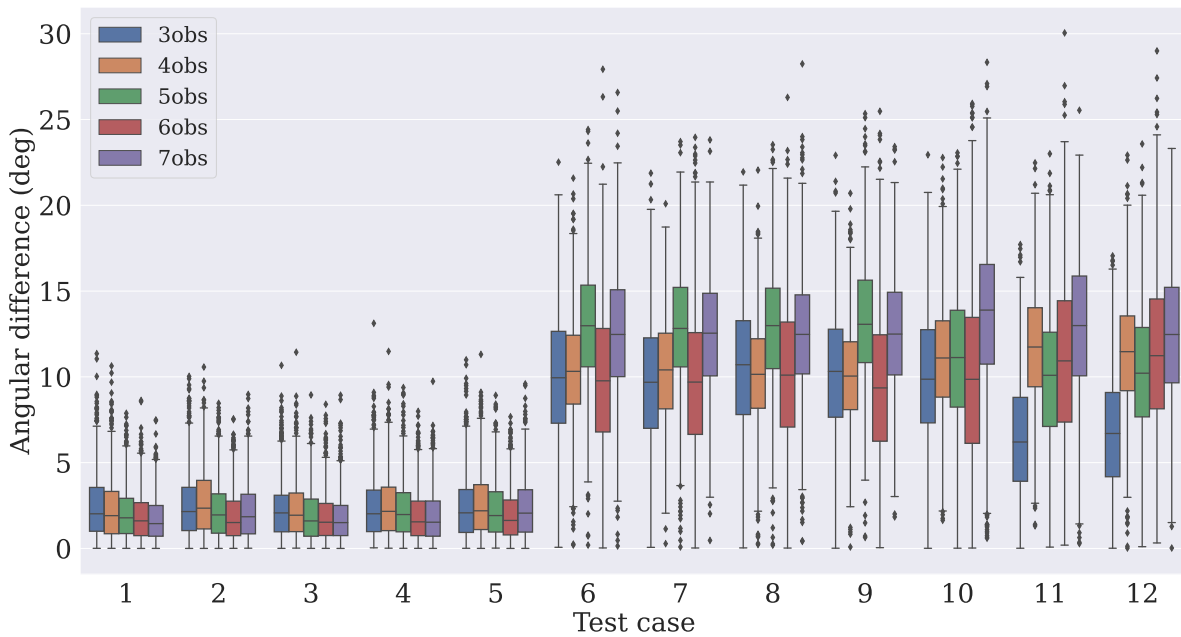
Figure 14: **Dispersion of the absolute angular difference when using a dropout rate of 10%.**



Source: Own authorship

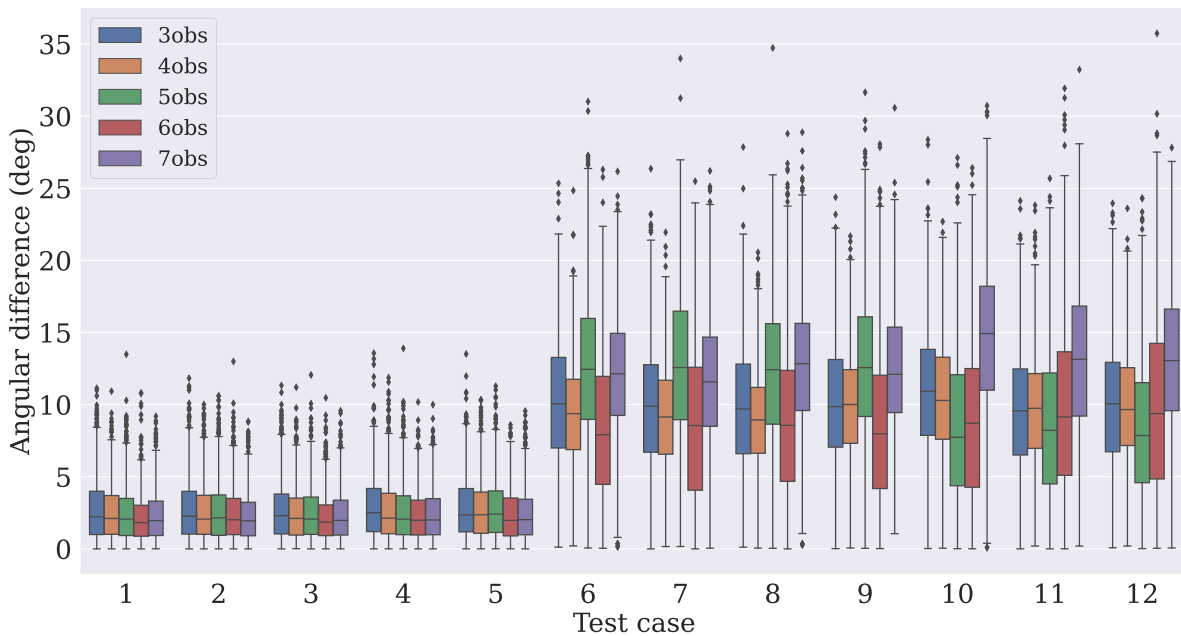
We verified the model trained with four vectors had the best results. However, to better understand the models, two new tests were performed. The first one evaluates the behavior of the models when using another dropout rate during the testing phase. And the other estimates how capable the models are for retaining knowledge when the number of switched-off neurons varies. For that matter, we evaluated the Wahba's error and the variance of the B-Swish-4obs model as previously performed, but sweeping the

Figure 15: Dispersion of the absolute angular difference when using a dropout rate of 15%.



Source: Own authorship

Figure 16: Dispersion of the absolute angular difference when using a dropout rate of 20%.

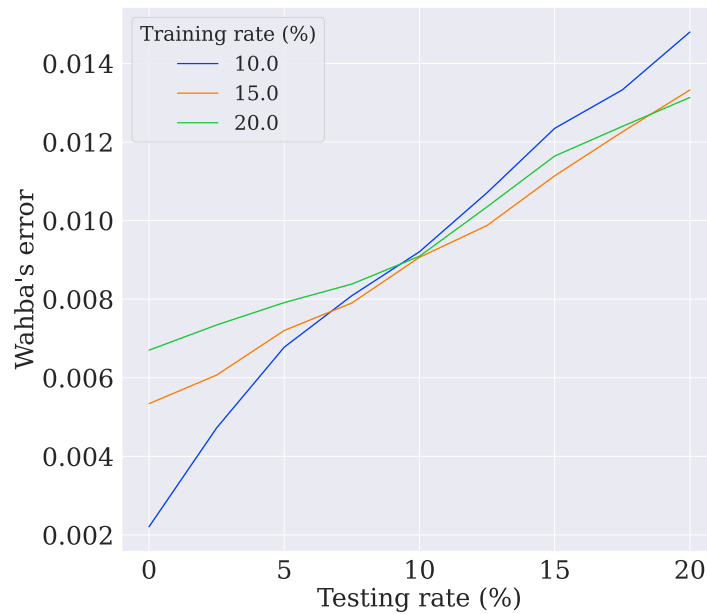


Source: Own authorship

dropout rate from 0% to 20%, with steps of 2.5%. From Figure, it is possible to notice the model trained with 10%, compared with the others, could not retain much knowledge when increasing the dropout rate, even though it achieved lower errors when the dropout

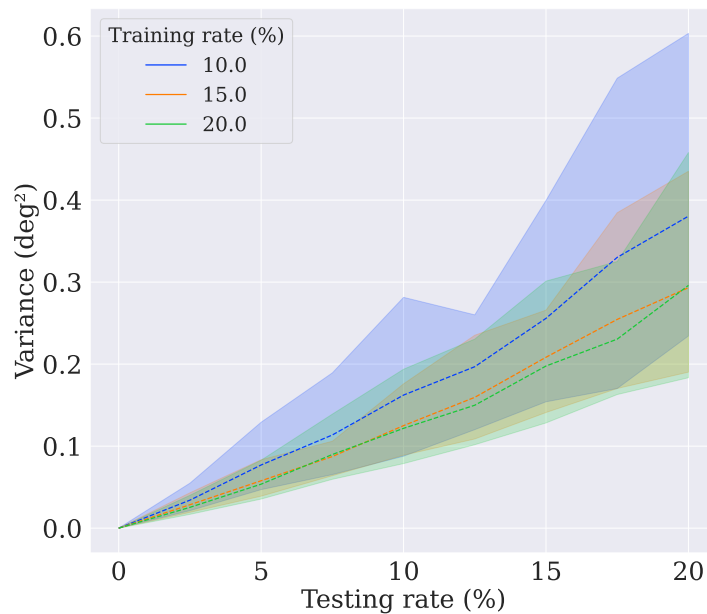
layer was disabled (0%). Similarly, the model trained with 10% was more uncertain of its predictions as the testing dropout rate increased, as displayed in Figure 18. The upper bound refers to the variance in the x -axis, the lower bound refers to the z -axis, and the dashed lines refer to the variance in the y -axis.

Figure 17: **Wahba's error evolution for Test Case 8.**



Source: Own authorship

Figure 18: **Variance evolution for Test Case 8.**



Source: Own authorship

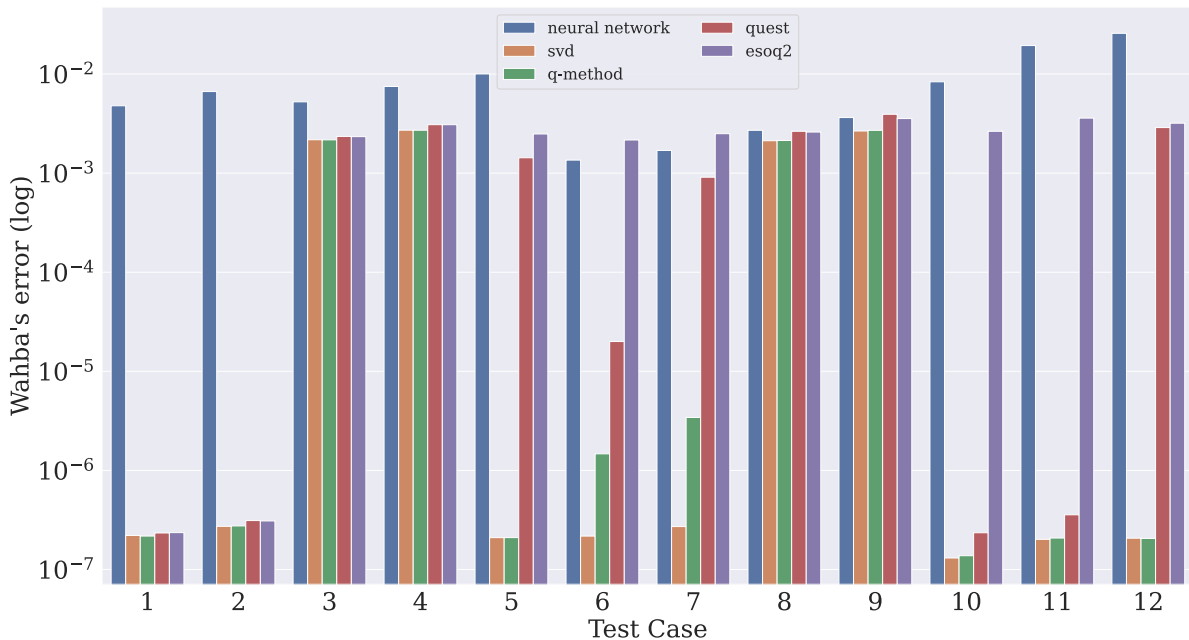
The results presented in the Figures 17 and 18 are only relevant if the model is running in an unstable environment, that is, it is expected that some neurons could "be

lost" during inference, thus the model has to be robust against the number of active neurons. However, we will compare the B-Swish-4obs trained with a dropout rate of 10%, our best scenario, with the traditional algorithms since it obtained the lowest Wahba's error when the dropout layers were disabled.

4.5 COMPARISON WITH TRADITIONAL ALGORITHMS

We carried out a benchmark using QUEST, SVD based, q-method, and ESOQ2 to verify whether the chosen neural network model was the most appropriate one for the problem in question. To do that, we considered the twelve test cases as previously defined. However, we generated a number of samples $\mathcal{N} = 4000$ for each case and applied their respective measurement noises $\mathcal{N}(0, \sigma_{\mathbf{i}})$. We calculated the Wahba's error for each sample fed into the algorithm by using the Equation (17), considering the $\mathbf{a}_{\mathbf{i}}$ given by each test case. After all samples had been assessed, we evaluated the mean of all errors $\sum_{\mathbf{j}=1}^{\mathcal{S}} L(\hat{\mathbf{R}}_{\mathbf{j}})$ for each case. We executed this procedure for all algorithms. The dropout layers were disabled during inference since there was no need for evaluating the model uncertainty in this phase. The results are depicted in Figure 19 on a logarithmic scale for better visualization.

Figure 19: Comparison of Wahba's error evaluation among some algorithms.



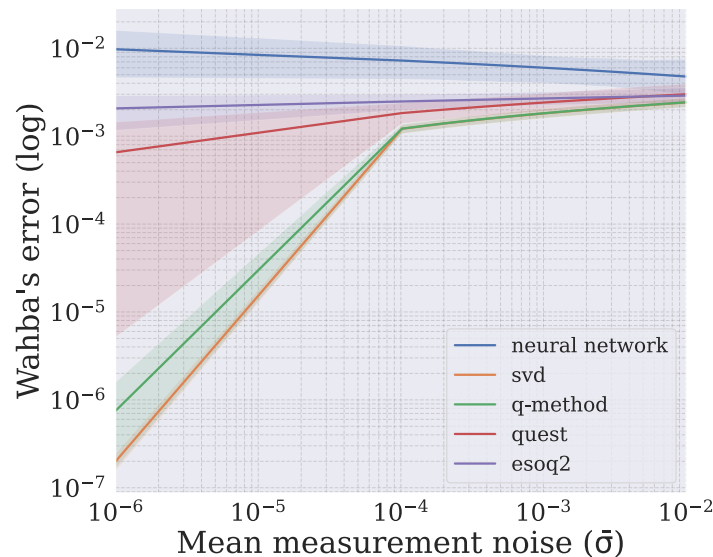
Source: Own authorship

As expected, SVD and q-Method algorithms achieved the lowest errors since they are the most robust. However, in some situations as cases, **3**, **4**, **8**, and **9**, which corrupt the input data with more significant noise, all algorithms obtained similar results. Although the

neural network has performed worse in most cases, their predictions were more constant, that is, the system error did not increase proportionally in the same way that occurred with the other algorithms when considerable noise was introduced to the input data.

The relationship between the measurement noise and Wahba's error is presented in Figure 20. This analysis considered a weighted average of the measurement noises for each test case. It was possible to notice that Wahba's error in the neural network remained constant when the weighted average increased, unlike traditional algorithms. Also, the presence of a single accurate observation vector (low valued σ) in a test case is enough to prevent the conventional algorithm from performing poorly. Case number 10 is an example of this statement since the traditional algorithms get less affected by the noise despite having two measurement noises with higher values. Due to that, we used Equation (49) to calculate the weights for the weighted average.

Figure 20: **Impact of average measurement noise over model performance.**



5 CONCLUSION

In this thesis, we proposed a neural network model, for the static attitude determination problem, based on an existing architecture called PointNet. We performed modifications in the system's layers, input, and output shapes due to poor results in preliminary tests. We noticed that changing the ReLU activation function to Swish resulted in an improvement in the loss over the validation set. The most noticeable gain happened when we changed the system input format. Before, we used two sets of vectors denoted as reference and body vectors, respectively, represented in distinct coordinate frames. By sending them in pairs, the model had to put more effort into understanding their relationship because there was no clear information about how they were related. On the other hand, the attitude profile matrix alleviated this work because it embeds the similarity between the vectors, that is, the cumulative distance between each axis of the reference and body vectors. It made the network converge quickly, in less than 2000 epochs.

Using a star tracking scenario during the training stage resulted in smaller Wahba's error values during the testing stage than using the actual weights. This is interesting because it removes the need for storing the actual weights \mathbf{a}_i to build the matrix attitude profile matrix \mathbf{B} . The error over the validation data indeed decreased when the number of measurement vectors increased. This did not reflect the actual behavior when presented to the test case scenarios. Maybe because the test cases used only up to three vectors and the models trained using more observation vectors could not generalize so well for fewer vectors. Furthermore, for the same test cases, the variances in the \mathbf{x} -axis changed abruptly from test 6 onwards. In these cases, all reference vectors have their boresight axis pointing in almost the same direction. In contrast, in the first cases, their boresight axes pointed to more spread-out directions. Therefore, the more similar their pointing axes, the noisier the profile matrix.

While the chosen model could not surpass traditional algorithms in the noisy cases, it did not suffer from it as the other algorithms did, achieving similar Wahba's error values. Therefore it is less sensitive to the input noise. As future work, we want to incorporate Bingham's distribution concepts into the model since our data type follows directional statistics. By doing so, it may be possible to estimate the uncertainty better, perhaps reducing its value and obtaining some improvements over the Wahba's loss.

REFERENCES

- BUKSHAT, Yannick; VETTER, Marcus. EfficientPose: An efficient, accurate and scalable end-to-end 6D multi object pose estimation approach. **arXiv preprint arXiv:2011.04307**, 2020.
- BULAT, Adrian; KOSSAIFI, Jean; TZIMIROPOULOS, Georgios; PANTIC, Maja. Toward fast and accurate human pose estimation via soft-gated skip connections. In: IEEE. 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020). [S.l.: s.n.], 2020. P. 8–15.
- CHANG, Angel X et al. Shapenet: An information-rich 3d model repository. **arXiv preprint arXiv:1512.03012**, 2015.
- CHIANG, Kai-Wei; CHANG, Hsiu-Wen; LI, Chia-Yuan; HUANG, Yun-Wen. An artificial neural network embedded position and orientation determination algorithm for low cost MEMS INS/GPS integrated sensors. **Sensors**, Molecular Diversity Preservation International, v. 9, n. 4, p. 2586–2610, 2009.
- CHOY, Christopher; DONG, Wei; KOLTUN, Vladlen. Deep global registration. In: PROCEEDINGS of the IEEE/CVF conference on computer vision and pattern recognition. [S.l.: s.n.], 2020. P. 2514–2523.
- CRAIG, John J. **Introduction to robotics: mechanics and control, 3rd Edition**. [S.l.]: Pearson Education International, 2005.
- CURTIS, W Dan; JANIN, Adam L; ZIKAN, Karel. A note on averaging rotations. In: IEEE. PROCEEDINGS of IEEE Virtual Reality Annual International Symposium. [S.l.: s.n.], 1993. P. 377–385.
- DAVENPORT, Paul B. A vector approach to the algebra of rotations with applications. National Aeronautics and Space Administration, v. 4696, 1968.
- DO, Thanh-Toan; CAI, Ming; PHAM, Trung; REID, Ian. Deep-6dpose: Recovering 6d object pose from a single rgb image. **arXiv preprint arXiv:1802.10367**, 2018.
- ESFAHANI, Mahdi Abolfazli; WANG, Han; WU, Keyu; YUAN, Shenghai. OriNet: Robust 3-D Orientation Estimation With a Single Particular IMU. **IEEE Robotics and Automation Letters**, IEEE, v. 5, n. 2, p. 399–406, 2019.

FRANASZEK, Marek; CHEOK, Geraldine S. Orientation uncertainty characteristics of some pose measuring systems. **Mathematical problems in engineering**, Hindawi, v. 2017, 2017.

FRIEDMAN, Milton. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the american statistical association**, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937.

GAL, Yarin; GHAHRAMANI, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: PMLR. INTERNATIONAL conference on machine learning. [S.l.: s.n.], 2016. P. 1050–1059.

GAO, Ge; LAURI, Mikko; ZHANG, Jianwei; FRINTROP, Simone. Occlusion resistant object rotation regression from point cloud segments. In: PROCEEDINGS of the European Conference on Computer Vision (ECCV). [S.l.: s.n.], 2018. P. 716–729.

GIBBS, Josiah Willard; WILSON, Edwin Bidwell. **Vector Analysis: A Text-book for the Use of Students of Mathematics and Physics, Founded Upon the Lectures of J. Willard Gibbs...** Yale University, New Haven, Connecticut: C. Scribner's sons, 1901. v. 11.

HASHIM, Hashim A. Attitude determination and estimation using vector observations: Review, challenges and comparative results. **arXiv preprint arXiv:2001.03787**, 2020.

HE, Kaiming; GKIOXARI, Georgia; DOLLÁR, Piotr; GIRSHICK, Ross. Mask r-cnn. In: PROCEEDINGS of the IEEE international conference on computer vision. [S.l.: s.n.], 2017. P. 2961–2969.

HORACHE, Sofiane; DESCHAUD, Jean-Emmanuel; GOULETTE, François. 3D Point Cloud Registration with Multi-Scale Architecture and Self-supervised Fine-tuning. **arXiv preprint arXiv:2103.14533**, 2021.

IWASE, Shun; LIU, Xingyu; KHIRODKAR, Rawal; YOKOTA, Rio; KITANI, Kris M. RePOSE: Real-Time Iterative Rendering and Refinement for 6D Object Pose Estimation. **arXiv preprint arXiv:2104.00633**, 2021.

KEAT, JE. Analysis of Least-Squares Attitude Determination Routine DOAOP (CSC/TM-77/6034). **Retrieved from Greenbelt**, 1977.

- LIU, Huajun; LIU, Fuqiang; FAN, Xinyi; HUANG, Dong. Polarized Self-Attention: Towards High-quality Pixel-wise Regression. **arXiv preprint arXiv:2107.00782**, 2021.
- LOSHCHILOV, Ilya; HUTTER, Frank. Decoupled weight decay regularization. **arXiv preprint arXiv:1711.05101**, 2017.
- MARKLEY, F Landis. Attitude Determination from Vector Observations: A Fast Optimal Matrix Algorithm. **The Journal of the Astronautical Sciences**, v. 41, n. 2, p. 261–280, 1993.
- MARKLEY, F Landis. Attitude determination using vector observations and the singular value decomposition. **Journal of the Astronautical Sciences**, v. 36, n. 3, p. 245–258, 1988.
- MARKLEY, F Landis. Unit quaternion from rotation matrix. **Journal of guidance, control, and dynamics**, v. 31, n. 2, p. 440–442, 2008.
- MARKLEY, F Landis; CRASSIDIS, John L. **Fundamentals of spacecraft attitude determination and control**. NASA Goddard Space Flight Center, Greenbelt, MD, USA/State University of New York, Amherst, NY, USA: Springer, 2014. v. 33.
- MARKLEY, F Landis; MORTARI, Daniele. How to estimate attitude from vector observations, 1999.
- MARKLEY, F Landis; MORTARI, Daniele. New developments in quaternion estimation from vector observations, 2000.
- MOAKHER, Maher. Means and averaging in the group of rotations. **SIAM journal on matrix analysis and applications**, SIAM, v. 24, n. 1, p. 1–16, 2002.
- MORTARI, Daniele. Second estimator of the optimal quaternion. **Journal of Guidance, Control, and Dynamics**, v. 23, n. 5, p. 885–888, 2000.
- NEMENYI, Peter Bjorn. **Distribution-free multiple comparisons**. [S.l.]: Princeton University, 1963.
- PAVLLO, Dario; FEICHTENHOFER, Christoph; GRANGIER, David; AULI, Michael. 3d human pose estimation in video with temporal convolutions and semi-supervised

- training. In: PROCEEDINGS of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2019. P. 7753–7762.
- PERETROUKHIN, Valentin; GIAMOU, Matthew; ROSEN, David M; GREENE, W Nicholas; ROY, Nicholas; KELLY, Jonathan. A smooth representation of belief over so (3) for deep rotation learning with uncertainty. **arXiv preprint arXiv:2006.01031**, 2020.
- POIESI, Fabio; BOSCAINI, Davide. Generalisable and distinctive 3D local deep descriptors for point cloud registration. **arXiv preprint arXiv:2105.10382**, 2021.
- QI, Charles R; SU, Hao; MO, Kaichun; GUIBAS, Leonidas J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2017. P. 652–660.
- RAMACHANDRAN, Prajit; ZOPH, Barret; LE, Quoc V. Searching for activation functions. **arXiv preprint arXiv:1710.05941**, 2017.
- RAMBACH, Jason R; TEWARI, Aditya; PAGANI, Alain; STRICKER, Didier. Learning to fuse: A deep learning approach to visual-inertial camera pose estimation. In: IEEE. 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). [S.l.: s.n.], 2016. P. 71–76.
- RODRIGUES, Olinde. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire. **Journal de Mathématiques Pures et Appliquées**, 1840.
- SHARF, Inna; WOLF, Alon; RUBIN, Miles B. Arithmetic and geometric solutions for average rigid-body rotation. **Mechanism and Machine Theory**, Elsevier, v. 45, n. 9, p. 1239–1251, 2010.
- SHUSTER, M. Approximate algorithms for fast optimal attitude computation. In: GUIDANCE and Control Conference. Palo Alto, CA, U.S.A.: [s.n.], 1978. P. 1249.
- SHUSTER, M D; OH, S D. Three-axis attitude determination from vector observations. **Journal of guidance and Control**, v. 4, n. 1, p. 70–77, 1981.

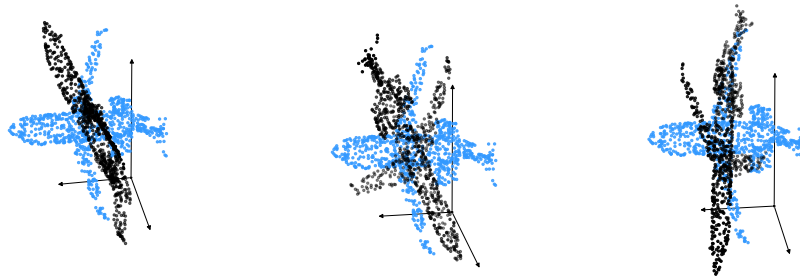
- SPRATLING, Benjamin B; MORTARI, Daniele. A survey on star identification algorithms. **Algorithms**, Molecular Diversity Preservation International, v. 2, n. 1, p. 93–107, 2009.
- SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.
- VALDENEbro, Angel G. Visualizing rotations and composition of rotations with the Rodrigues vector. **European Journal of Physics**, IOP Publishing, v. 37, n. 6, p. 065001, 2016.
- WAHBA, Grace. A least squares estimate of satellite attitude. **SIAM review**, SIAM, v. 7, n. 3, p. 409–409, 1965.
- WEBER, Daniel; GÜHMANN, Clemens; SEEL, Thomas. Neural Networks Versus Conventional Filters for Inertial-Sensor-based Attitude Estimation. **arXiv preprint arXiv:2005.06897**, 2020.
- WERTZ, James R. **Spacecraft attitude determination and control**. Microcosm Inc., Torrance, CA: Springer, 1978. v. 73. DOI: 10.1007/978-94-009-9907-7.
- XIANG, Yu; SCHMIDT, Tanner; NARAYANAN, Venkatraman; FOX, Dieter. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. **arXiv preprint arXiv:1711.00199**, 2017.
- YANG, Yaguang. **Spacecraft Modeling, Attitude Determination, and Control: Quaternion-Based Approach**. Rockville, Maryland, USA: CRC Press, 2019.
- ZAKHAROV, Sergey; SHUGUROV, Ivan; ILIC, Slobodan. Dpod: 6d pose object detector and refiner. In: PROCEEDINGS of the IEEE/CVF International Conference on Computer Vision. [S.l.: s.n.], 2019. P. 1941–1950.
- ZHANG, GUANGJUN. **STAR IDENTIFICATION: Methods, Techniques and Algorithms**. Beihang University, Beijing, China: Springer, 2019.

ZHOU, Yi; BARNES, Connelly; LU, Jingwan; YANG, Jimei; LI, Hao. On the continuity of rotation representations in neural networks. In: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2019. P. 5745–5753.

APPENDIX A – EVALUATION ON SHAPENET

We executed some other tests to evaluate how the proposed model behaves on the ShapeNet (CHANG et al., 2015) dataset, where two versions of the proposed model were tested. The first one considers the mapping function proposed by (ZHOU et al., 2019) as the output of the model. The second one considers the quaternion estimation from a symmetric matrix as proposed by (PERETROUKHIN et al., 2020). In both architectures we used the attitude profile matrix as input to the model. By doing that, we tested whether this new input format was capable of boosting the models’ performance on the pose estimation task.

Figure 21: **A sample from the ShapeNet airplane category rotated by three different rotation matrices.**



Source: Own authorship

We recreated the experiment from (ZHOU et al., 2019) on 2,290 airplane point clouds¹ from ShapeNet airplane category, with 400 testing point clouds. At training time, we transformed the data by rotating each sample in the dataset with 10 random rotation matrices. Figure 21 displays a sample from the dataset (in blue) rotated by three different DCMs (in black). During the training phase, we split 4% of the point clouds into a validation dataset. At testing time, we applied 100 random rotations to each of the testing point clouds.

The models described in (ZHOU et al., 2019) and (PERETROUKHIN et al., 2020), which we will call **6D** and **A**, respectively, consume raw point clouds to predict a valid rotation representation. However, we noticed these models could be more sensitive to the number of points used in each sample. Therefore, we trained each model considering 50 and 1200 points to validate their performances. We trained the models for 2000 epochs

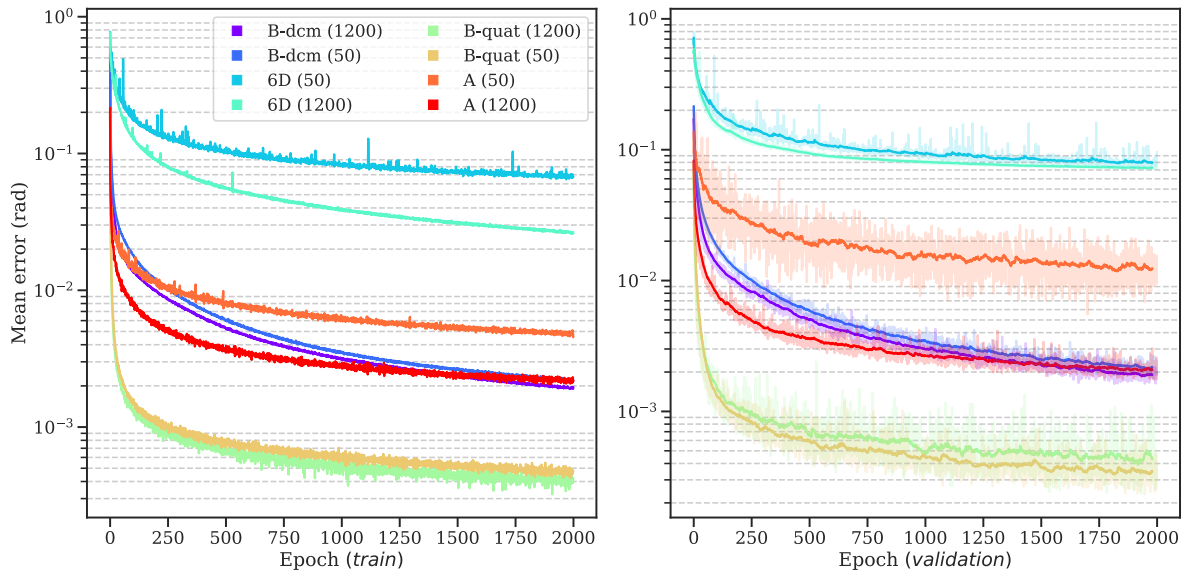
¹ Each point cloud in the dataset is composed of a collection of 3D coordinates. Thus, each coordinate corresponds to a point from the point cloud.

using the Adam optimizer with a learning rate $\mathbf{lr} = 10^{-4}$, and a weight decay of 10^{-4} being applied every 500 epochs. The \mathbf{lr} was also reduced by a factor of ten every 500 epochs. The models that used the mapping function $\mathbf{f} : \mathcal{R} \rightarrow \mathcal{X}$ as output were trained considering the loss function defined by Equation (40). The other ones, which predict quaternions, were trained using the loss function defined by Equation (52). Figure 22 shows the mean angular distance between the groundtruth and predicted rotations after 2000 epochs.

$$d_{quat}(\mathbf{q}, \hat{\mathbf{q}}) = \min(\|\mathbf{q} - \hat{\mathbf{q}}\|_2, \|\mathbf{q} + \hat{\mathbf{q}}\|_2) \quad (52a)$$

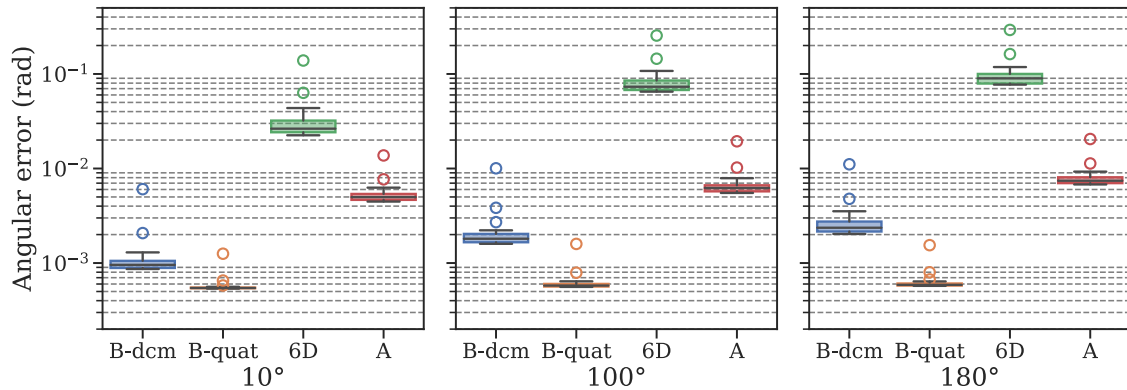
$$\phi(\mathbf{q}, \hat{\mathbf{q}}) = 2d_{quat}^2(4 - d_{quat}^2) \quad (52b)$$

Figure 22: Mean angular error during training and validation.



Source: Own authorship

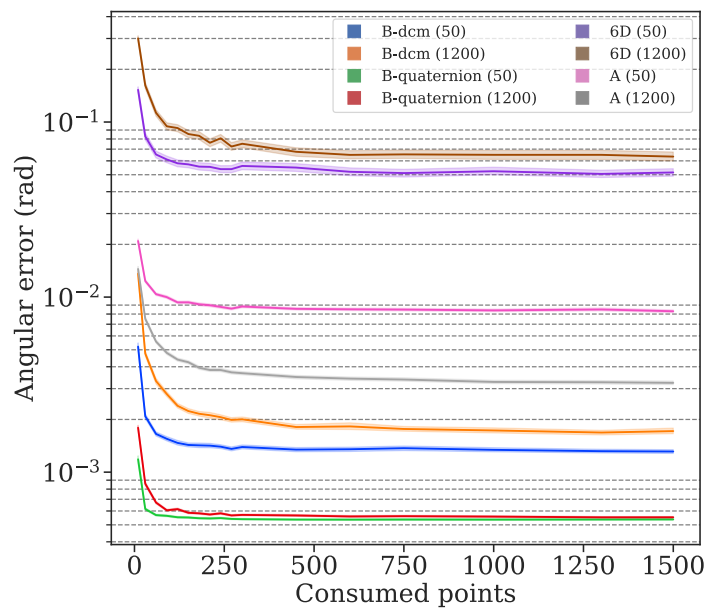
Then, we also evaluated their performances for specific rotation angles: 10° , 100° and 180° . For this test, we generated 100 random rotation matrices with fixed angle values for each of the 400 testing point clouds. We also considered different quantities of consumed points: 10, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 450, 600, 750, 1000, 1300, and 1500. In this way, we could evaluate the overall capabilities of the models, where the angular distance between the groundtruth and predicted rotation representations were measured, as displayed in Figure 23.

Figure 23: Overall performance for three different ϕ_{max} .

Source: Own authorship

Finally, we estimated the angular distance between the groundtruth and predicted rotation representations for angles ranging from $[0^\circ, 180^\circ]$ also considering different quantities of consumed points as previously performed. Figure 24 displays their results. As expected, our model showed more robustness against the number of consumed points, and greatly improved the results on the ShapeNet dataset. Figure 25 displays the proposed architectures used in the tests. The first convolution layer uses kernels of size 1, and the second uses kernels of size 9.

Figure 24: Evolution of the angular error when varying the number of consumed points.



Source: Own authorship

Figure 25: Model architectures, where a) was built considering the mapping function $6D \rightarrow SO(3)$, and b) considering the eigenvector extraction from a symmetric matrix.

